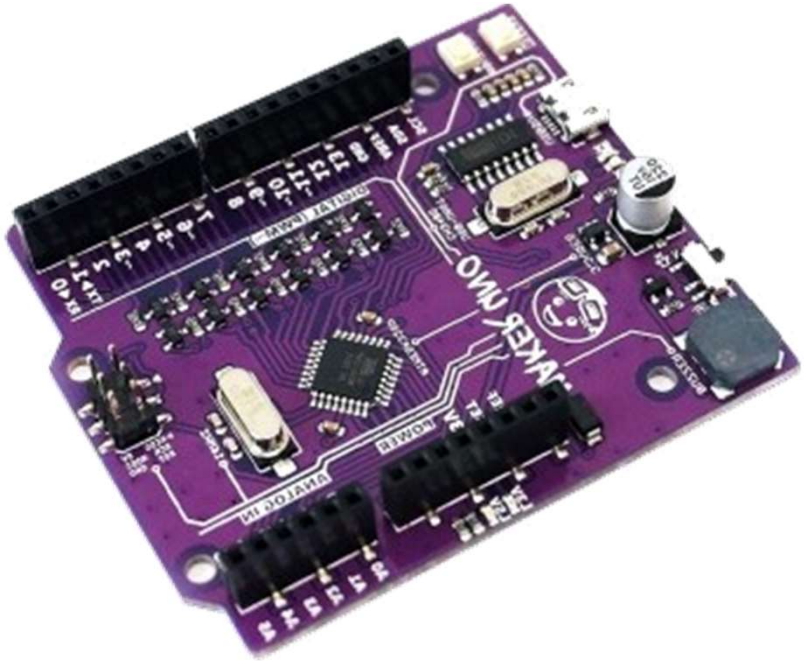
Arduino Essential Training v8



Trainer: Shahul H. Maricar

Website:www.tertiarycourses.com.sg
Email: enquiry@tertiaryinfotech.com

About the Trainer

Shahul H. Maricar has been a content developer and webmaster, building educational websites and applications with HTML, CSS and JavaScript. He then served as an IT analyst, writing programs for automating custom workflows as well as data extraction and analysis in the healthcare field.

He is currently a freelance educator and is actively involved with development projects in game programming, computer-aided design and computer graphics.

Let's Know Each Other...

Say a bit about yourself
- Name
- What Industry you are from?
- Do you have any prior knowledge in Arduino?
- Why do you want to learn Arduino?
- What do you expect to learn from this course?

Ground Rules

- Set your mobile phone to silent mode
- Actively participate in the class. No question is stupid.
- Respect each other view
- Exit the class silently if you need to step out for phone call, toilet break

Ground Rules for Virtual Training

- Upon entering, mute your mic and turn on the video. Use a headset if you can
- Use the 'raise hand' function to indicate when you want to speak
- Participant actively. Feel free to ask questions on the chat whenever.
- Facilitators can use breakout rooms for private sessions.

Guidelines for Facilitators

1. Once all the participants are in and introduce themselves
2. Goto gallery mode, take a snapshot of the class photo - makes sure capture the date and time
3. Start the video recording (only for WSQ courses)
4. Continue the class
5. Before the class end on that day, take another snapshot of the class photo - makes sure capture the date and time
6. For NRIC verification, facilitator to create breakout room for individual participant to check (only for WSQ courses)
7. Before the assessment start, take another snapshot of the class photo - makes sure capture the date and time (only for WSQ courses)
8. For Oral Questioning assessment, facilitator to create breakout room for individual participant to OQ (only for WSQ courses)
9. End the video recording and upload to cloud (only for WSQ courses)
10. Assessor to send all the assessment records, assessment plan and photo and video to the staff (only for WSQ courses).

Prerequisite

This is a beginner course. No prior knowledge is assumed.

Agenda

Topic 1 Introduction of Embedded Systems and Arduino

- Overview of Embedded Systems
- Applications of Embedded Systems
- Introduction to Arduino Microcontroller
- Serial Communication

Topic 2 Introduction to Basic Electronics

- Basic Electronics Concepts
- Digital Input/Output
- Pulse Width Modulation (PWM)

Topic 3 Analog Sensors and Transducers

- Overview of Analog Sensors and Transducers
- Analog Input/Output
- Servo Control

Topic 4 Digital Sensors and Transductors

- Overview of Digital Sensors and Transducers
- Light Activation by Motion Detection

Agenda

Topic 5 Actuators

- Overview of Actuator Networks
- Motor Control

Final Assessment

- Practical Performance
- Oral Questioning

Google Classroom

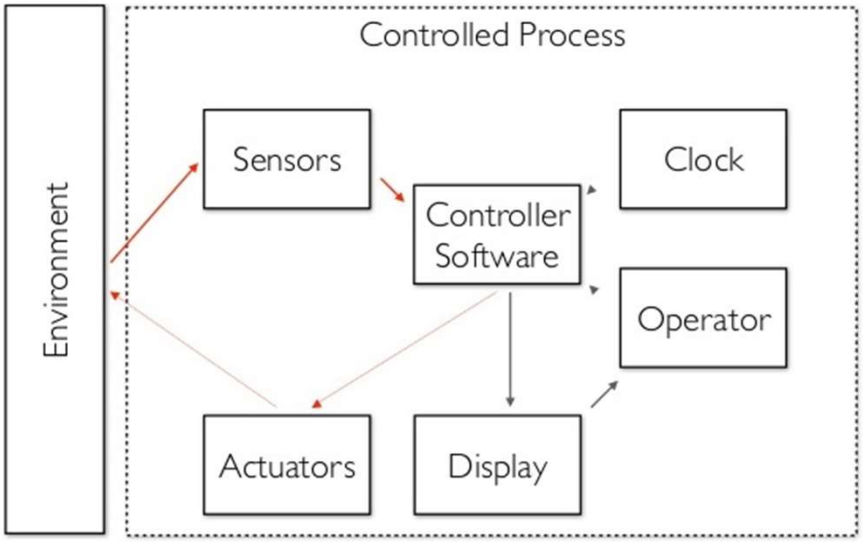
- The resources can be found on the Google classroom
- Goto https://classroom.google.com and enter the class code below
- If you have problem to access the Google Classroom, please inform trainer or the staff

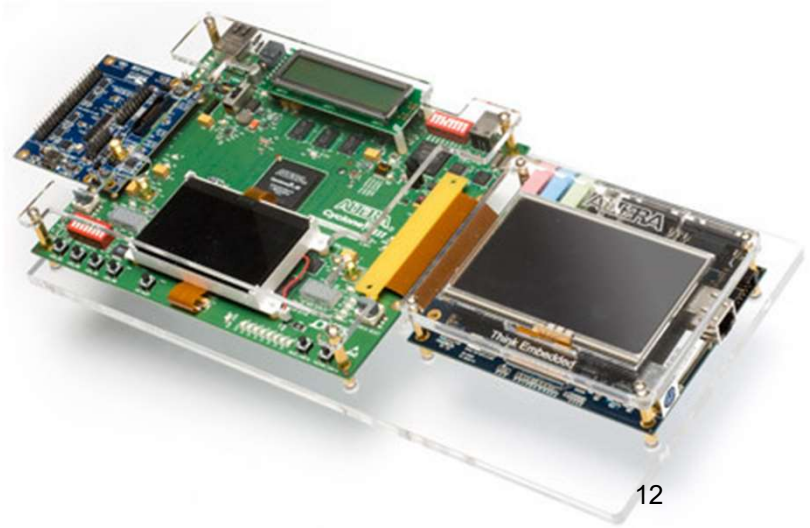
lc57ali

Topic 1
Introduction of Embedded Systems and Arduino

Overview of a Embedded System

- An embedded system is a microprocessor-based computer hardware system with software that is designed to perform a dedicated function, either as an independent system or as a part of a large system.
- Embedded systems are managed by microcontrollers or digital signal processors (DSP), application-specific integrated circuits (ASIC), field-programmable gate arrays (FPGA)
- Embedded systems programming instructions, referred to as firmware, are stored in read-only memory or flash memory chips, running with limited computer hardware resources.
- Embedded systems connect with the outside world through peripherals, linking input and output devices.





Structure of an Embedded System

- The basic structure of an embedded system includes the following components:
  - **Sensor:** The sensor measures and converts the physical quantity to an electrical signal, which can then be read by an embedded systems engineer or any electronic instrument. A sensor stores the measured quantity to the memory.
  - **A-D Converter:** An analog-to-digital converter converts the analog signal sent by the sensor into a digital signal.
  - **Processor & ASICs:** Processors assess the data to measure the output and store it to the memory.
  - **D-A Converter:** A digital-to-analog converter changes the digital data fed by the processor to analog data
  - Actuator: An actuator compares the output given by the D-A Converter to the actual output stored and stores the approved output.

Applications of Embedded Systems

Industrial Robots

GPS Receivers

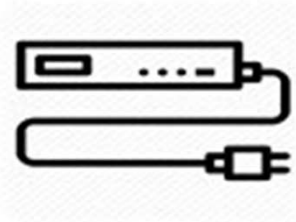
Digital Cameras

DVD Players

Wireless Routers

Embedded Systems

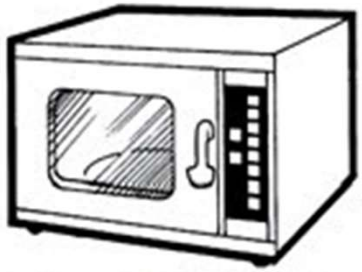
MP3 Players



Set top Boxes

Gaming Consoles

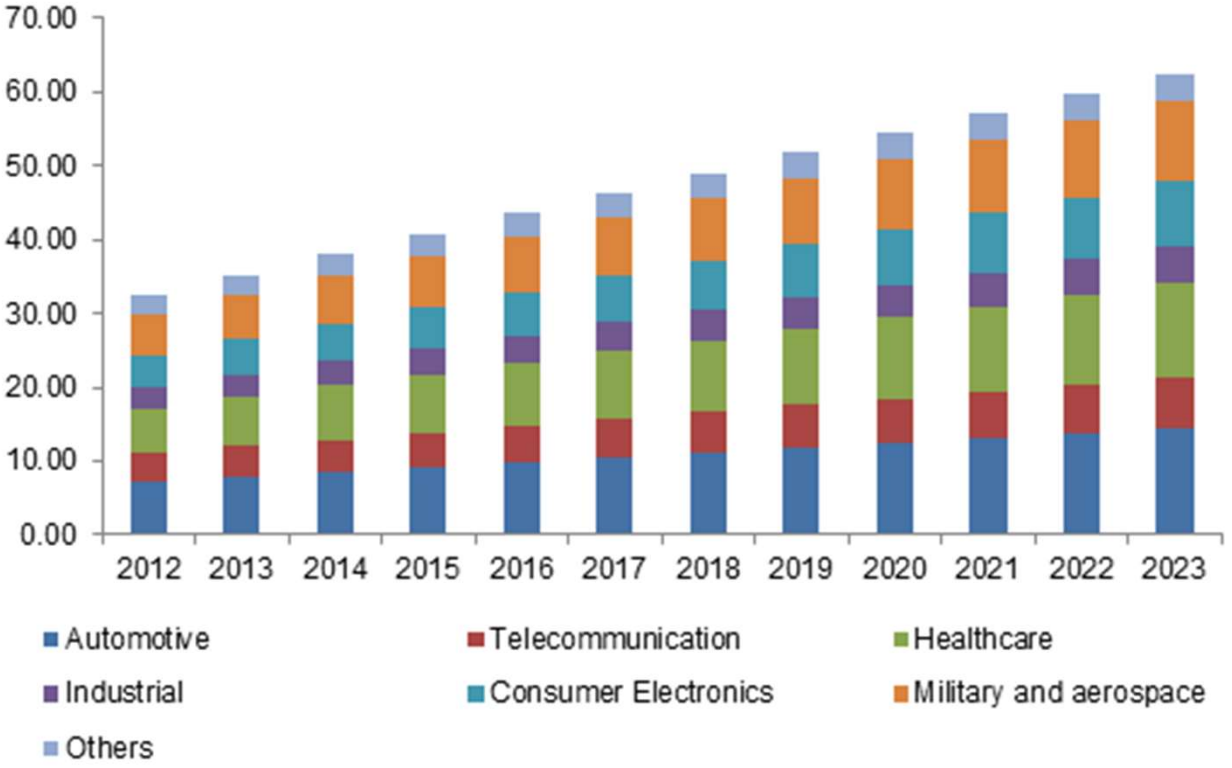
Photocopiers



Microwave Ovens

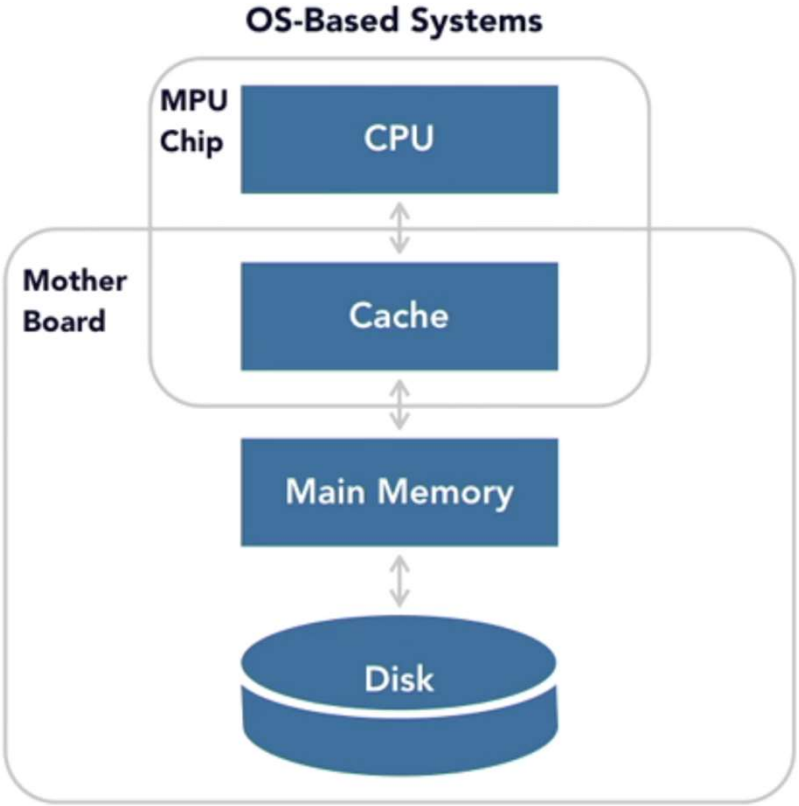
Future Trend of Embedded Systems

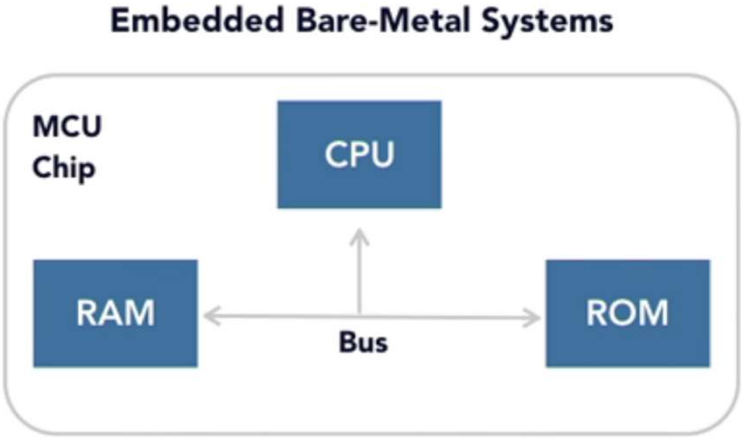
- The industry for embedded systems is expected to continue growing rapidly, driven by the continued development of Artificial Intelligence (AI), Virtual Reality (VR) and Augmented Reality (AR), machine learning , deep learning, and the Internet of Things (IoT).
- According to a 2018 report published by QYResearch, the global market for the embedded systems industry was valued at $68.9 billion in 2017 and is expected to rise to $105.7 billion by the end of 2025.



Desktop vs Embedded Systems

Desktop/Mobile	Embedded System
• General Purpose • Over 1MB of Memory • High performance CPU • Unlimited Energy • Goal: Cost/Performance	• Digital Control of hardware • Less than 2kB of Memory • Low end microcontroller • Battery operated • Goal: reach the necessary performance at the lowest cost





Memory in Embedded Systems

- Memory is typically limited in Embedded System due to the following reasons
  - Digital Control is fine with a few variables
  - Use less area in a chip
  - Cost less
  - Use less energy
  - Help keep addresses short (16 bit)

Storage in Embedded Systems

- Storage is typically limited in Embedded System due to the following reasons
  - Digital Control uses simple algorithms
  - Use less area in a chip
  - Cost less
  - Use less energy
  - Help keep addresses short (16 bit)

Power in Embedded Systems

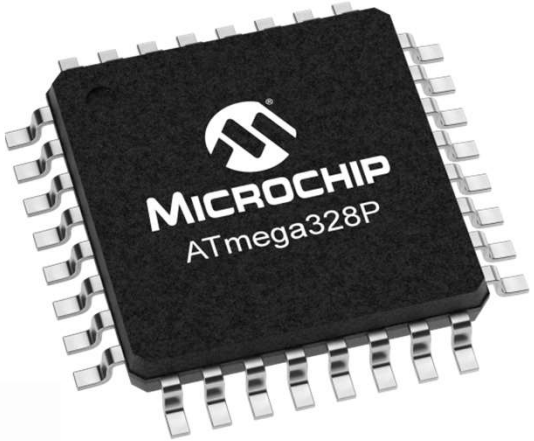
- Power is typically limited in Embedded System due to the following reasons:
  - Devices are usually battery operated
  - Recharging/replacing battery could be a problem
  - AC powered embedded devices are on all day
  - Embedded MCUs rarely have heat sinks

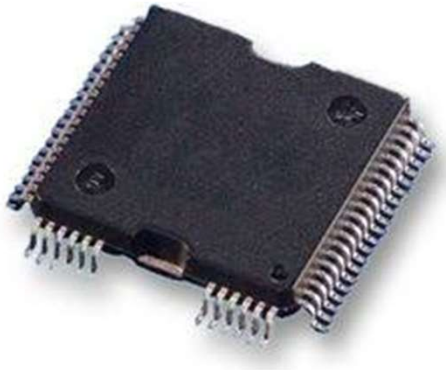
Low End MCU in Embedded Systems

- Embedded System typically use low cost and low end microcontroller (MCU) for the following reasons:
  - They consume less energy
  - Digital control uses simple algorithms
  - They cost less
  - Memory is small anyway

Commercial Microcontrollers (MCU)

- ATmega328P by MicroChip: 8 bit
- S08 architecture by NXP: 8 bit
- MSP architecture by Texas Instrument: 8 bit
- ARM Cortex-M: 32 bit





ARM Cortex Microcontroller

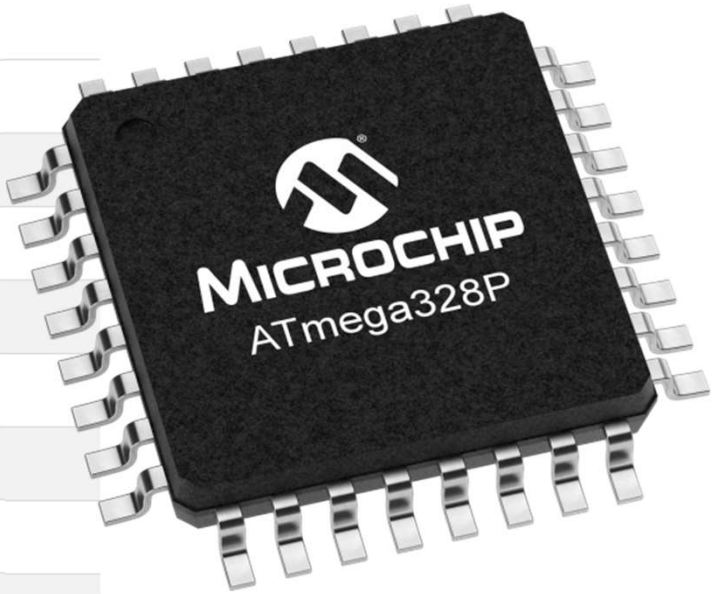
- Arm® Cortex®-M4 MCU
- 12kB SRAM
- 64kB Flash Memory

source: https://www.st.com/en/microcontrollers-microprocessors/stm32-32-bit-arm-cortex-mcus.html

ATmega328P Microcontroller

Parametrics

Name	Value
Program Memory Type	Flash
Program Memory Size (KB)	32
CPU Speed (MIPS/DMIPS)	20
SRAM (B)	2,048
Data EEPROM/HEF (bytes)	1024
Digital Communication Peripherals	1-UART, 2-SPI, 1-I2C
Capture/Compare/PWM Peripherals	1 Input Capture, 1 CCP, 6PWM
Timers	2 x 8-bit, 1 x 16-bit
Number of Comparators	1
Temperature Range (°C)	-40 to 85
Operating Voltage Range (V)	1.8 to 5.5
Pin Count	32
Low Power	Yes



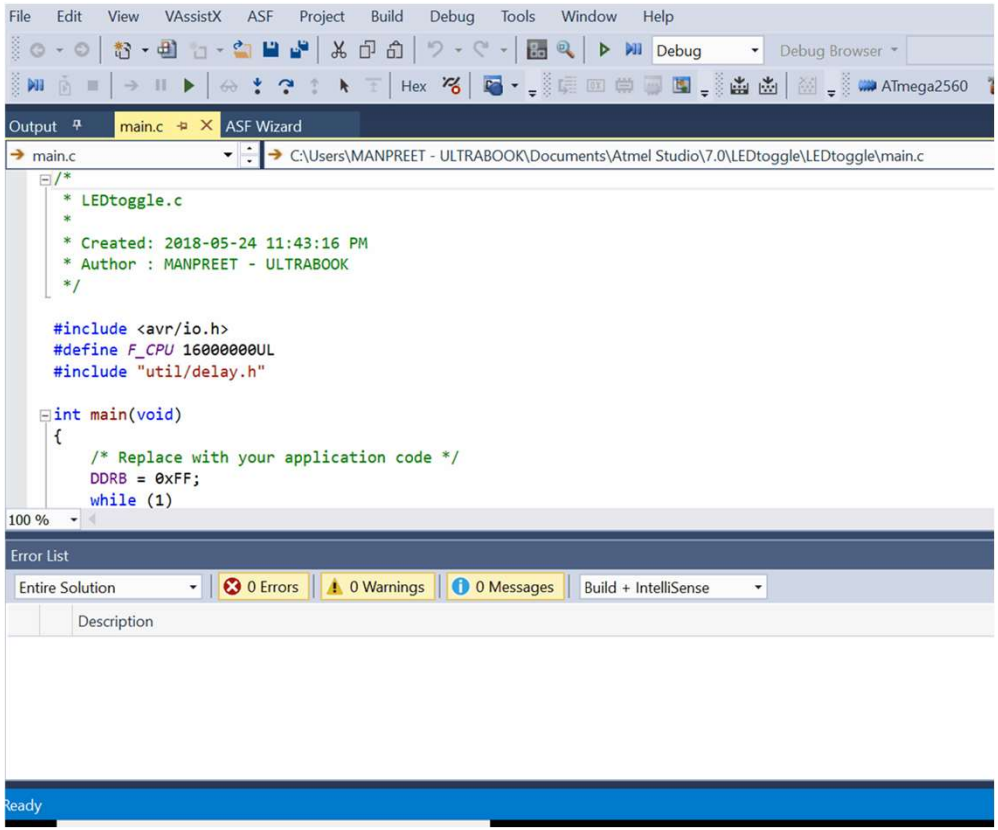
source: https://www.microchip.com/wwwproducts/en/ATmega328p

Embedded C Code General Structure

/* comments */

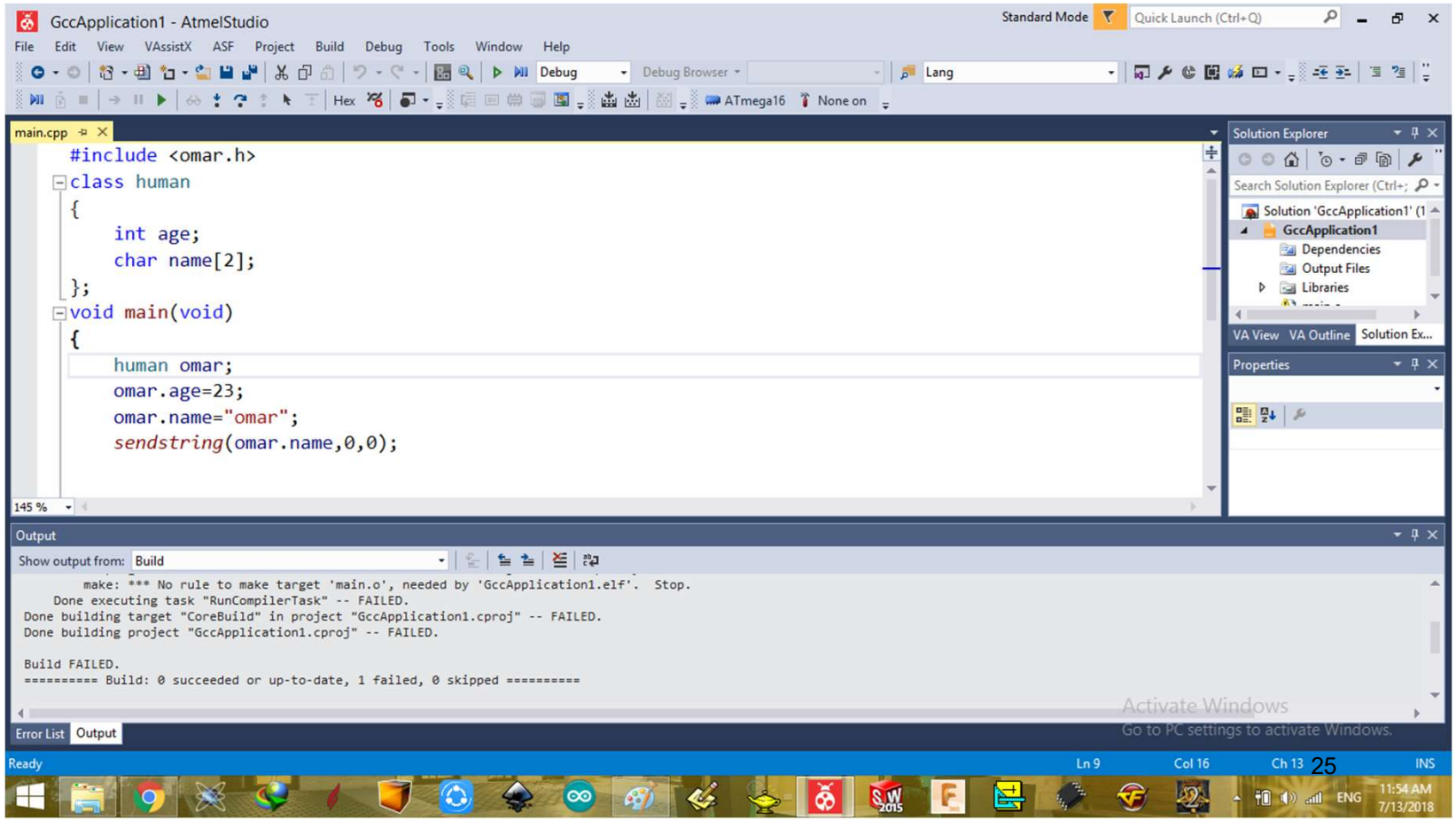
preprocessor directives
global variables

main() function
{
local variables
statements
.....
.....
}



Atmel Studio

- The Atmel Studio 7 IDP gives you a seamless and easy-to-use environment to write, build and debug your applications written in C/C++ or assembly code for ATmega328.
- Studio 7 can also seamlessly import your Arduino sketches as C++ projects, providing a simple transition path from Makerspace to Marketplace.

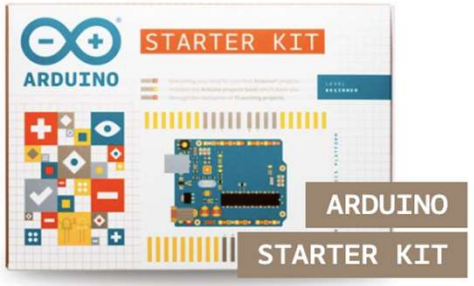


Arduino Embedded System

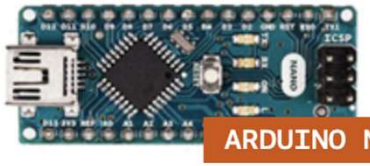
- Arduino (https://www.arduino.cc/) is one of those Embedded System, known as embedded Development Board, which got very famous in the maker's community due to its free and open source nature.
- It can be used for making any kind of simple automated electronic projects
- It consist of
  - ATmega328P chip
  - General Purpose Input/Output (GPIO)
  - Timers
  - Serial Ports
  - Analog-to-Digital Converters

Arduino Products - Entry Level

ARDUINO UNO



ARDUINO STARTER KIT



ARDUINO NANO

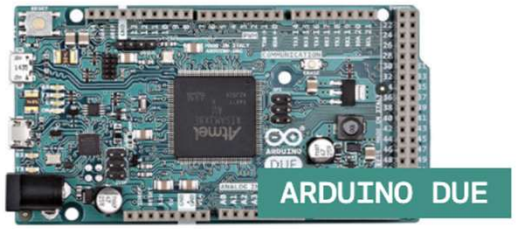
ARDUINO LEONARDO

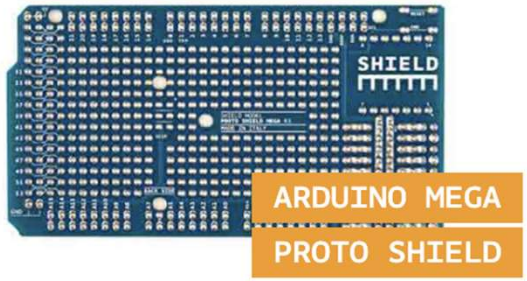
ARDUINO MICRO

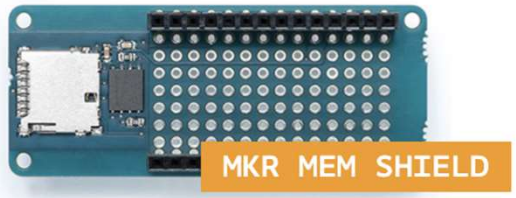
ARDUINO NANO EVERY

MKR2UNO ADAPTER

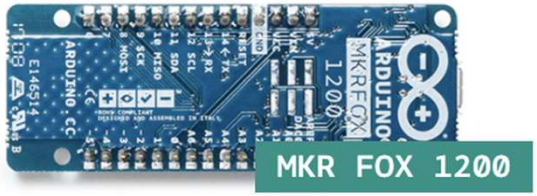
Arduino Products - Enhanced Level

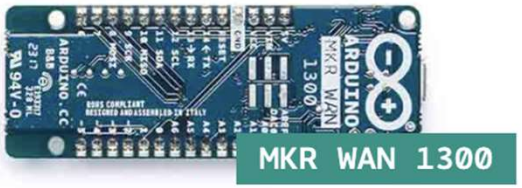


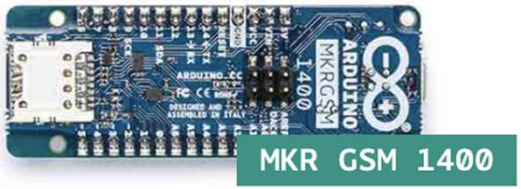


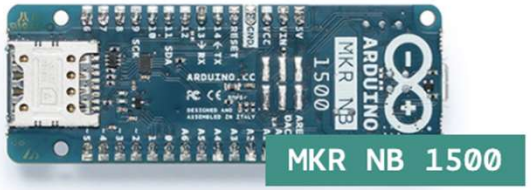


Arduino Products for IoT

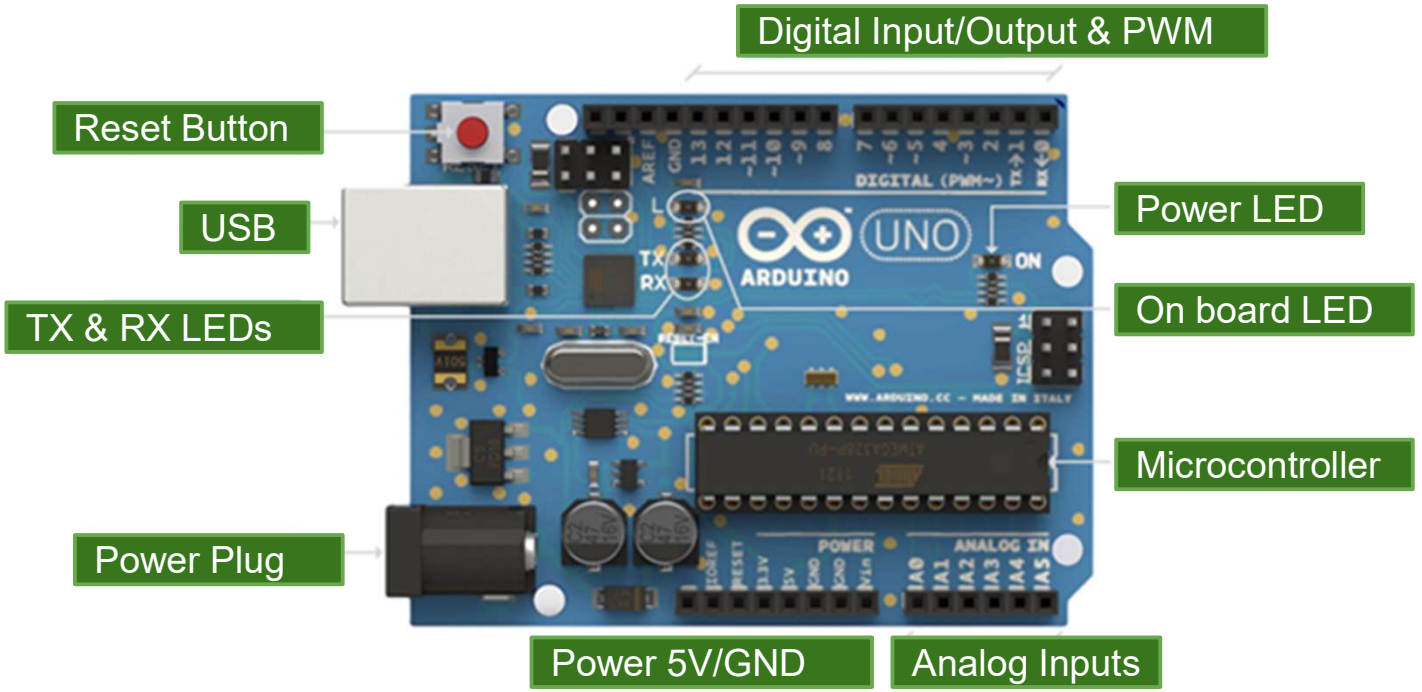








Arduino UNO Layout



Microcontroller: ATmega328 Operating Voltage: 5V

Digital I/O Pins: 14 (6 PWM output) Analog Input Pins : 6

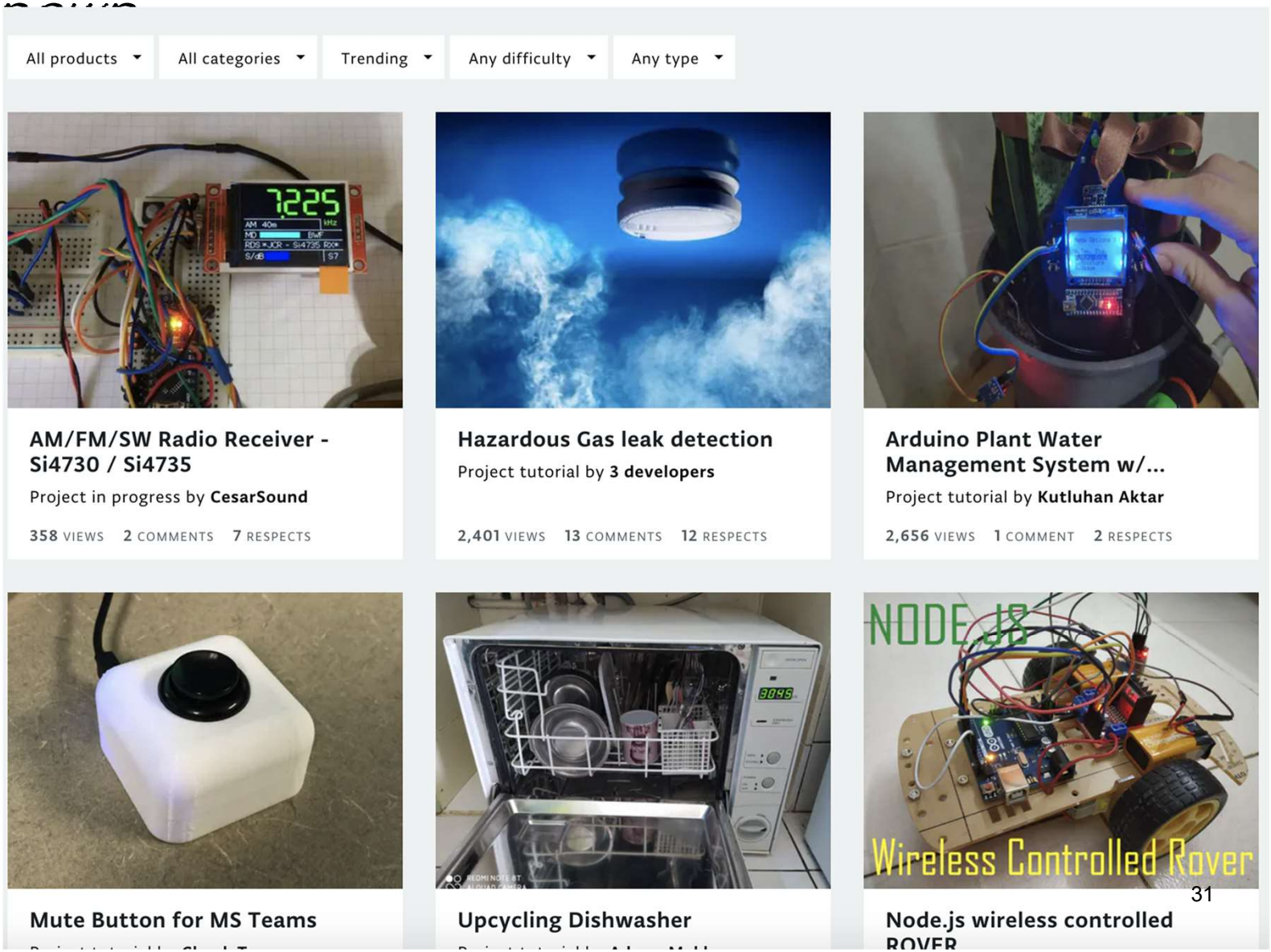
Flash Memory: 32 KB (ATmega328) Clock Speed: 16 MHz

PWM: 3, 5, 6, 9, 10, and 11. Provide 8-bit PWM output with the analogWrite() function.

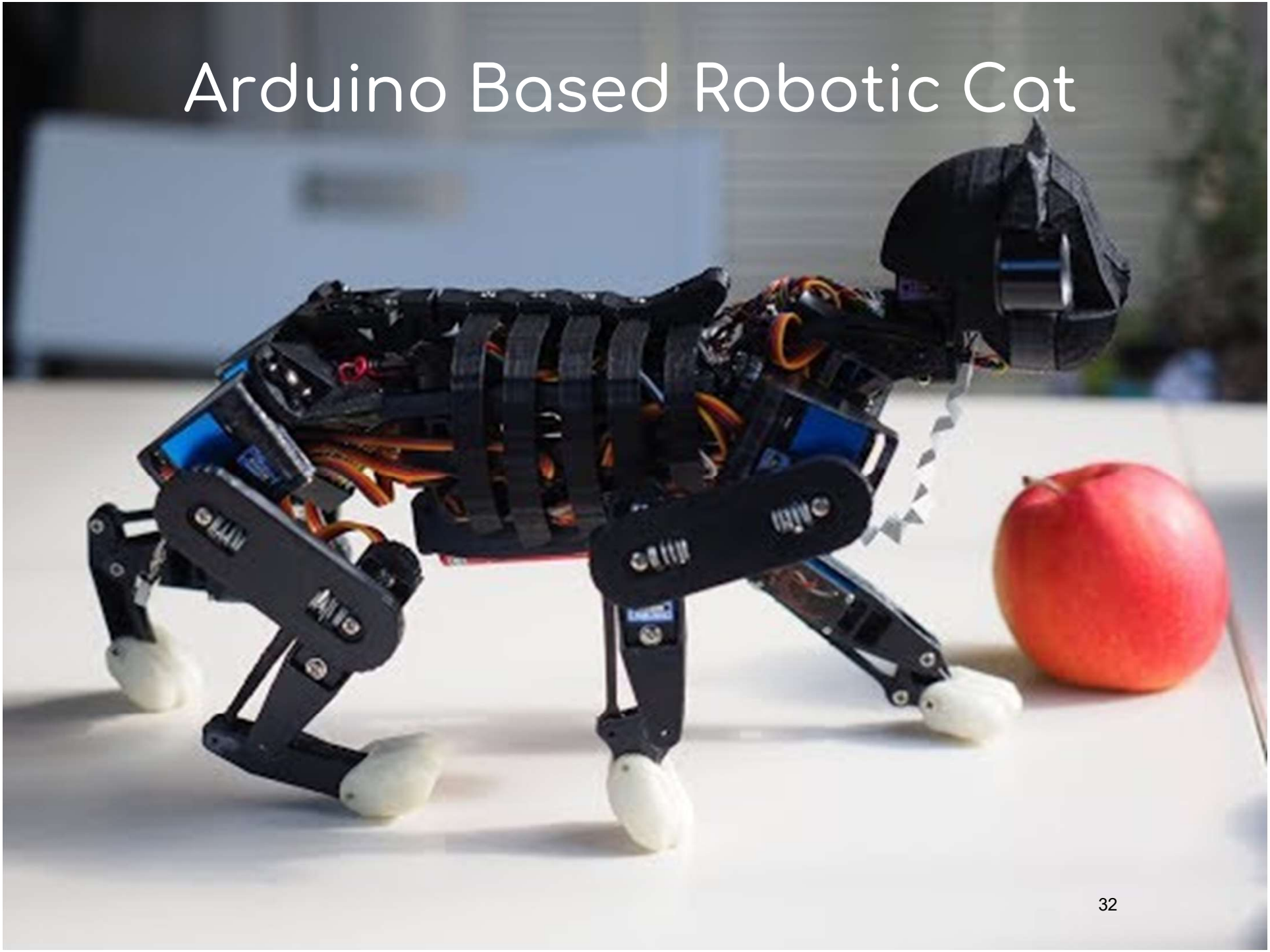
LED: 13. There is a built-in LED connected to digital pin 13.

Arduino Project Hub

- Arduino is one of those Embedded System, known



Arduino Based Robotic Cat



Arduino Based Follow Me Cooler

History of Arduino

- Arduino was started in 2005 by Massimo Banzi and other co-founders from Interaction Design Institute Ivrea (IDII) in Ivrea, Italy.
- It was named after the bar where the co-founders visited often. The name of the bar is named after King Arduino.

Father of Arduino

Embedded C Programming

- Embedded C language is most frequently used to program the microcontroller.
- Embedded C is a generic term given to a programming language written in C, which is associated with a particular hardware architecture.
- Embedded C is an extension to the C language with some additional header files. These header files may change from controller to controller.

Coding Embedded C on Arduino

Arduino provides two ways to code the embedded C programs

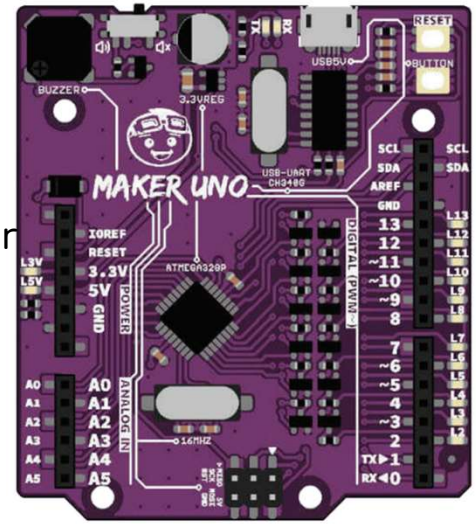
- Offline: http://www.arduino.cc/download
- Online: https://create.arduino.cc/

- Maker UNO is an Arduino Compatible board specially designed to simplify building your projects
- You can share the same library and code. It has 12x LEDs, 1x piezo buzzer and 1x programmable button on the Maker UNO.
- The DC jack power input socket as most of the boards used in classes are powered using USB.
- The ATmega16u2 with CH340 to bring down the cost.

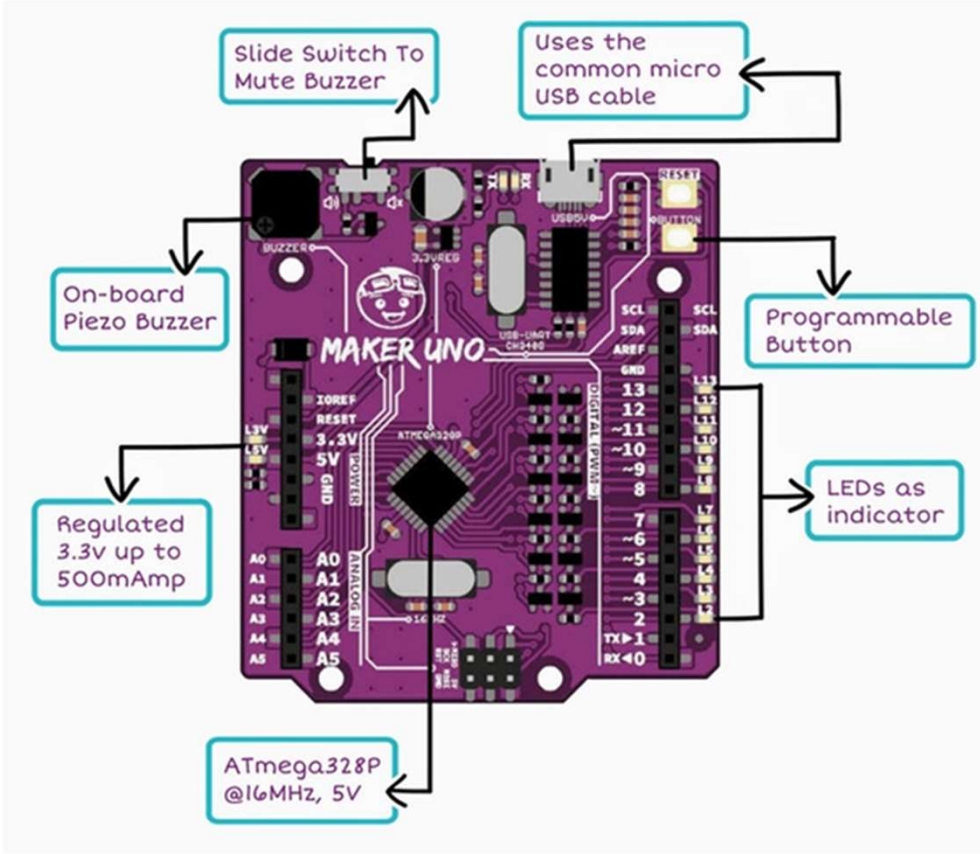
Maker UNO

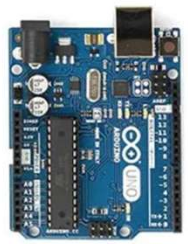
- Maker UNO, an Arduino UNO compatible board designed and developed specially for students to learn coding and microcontroller.

MAKER - UNO Features:
- SMD ATmega328P microcontroller(the same microcontroller on Arduino UNO) with Optiboot (UNO) Bootloader.
- USB Programming facilitated by the CH340.
- Input voltage: USB 5V, from computer, power bank or standard USB adapter.
- 500mA (maximum) 3.3V voltage regulator.
- 0-5V outputs with 3.3V compatible inputs.
- 14 Digital I/O Pins (6 PWM outputs).
- 6 Analog Inputs.
- ISP 6-pin Header.
- 32k Flash Memory.
- 16MHz Clock Speed.
- R3 Shield Compatible.
- LED array for 5V, 3.3V, TX, RX and all digital pins.
- On board programmable push button (pin 2, need to congure as INPUT_PULLUP).
- On board piezo buzzer (pin 8).
- Utilize USB Micro-B socket.



Maker UNO vs Arduino UNO

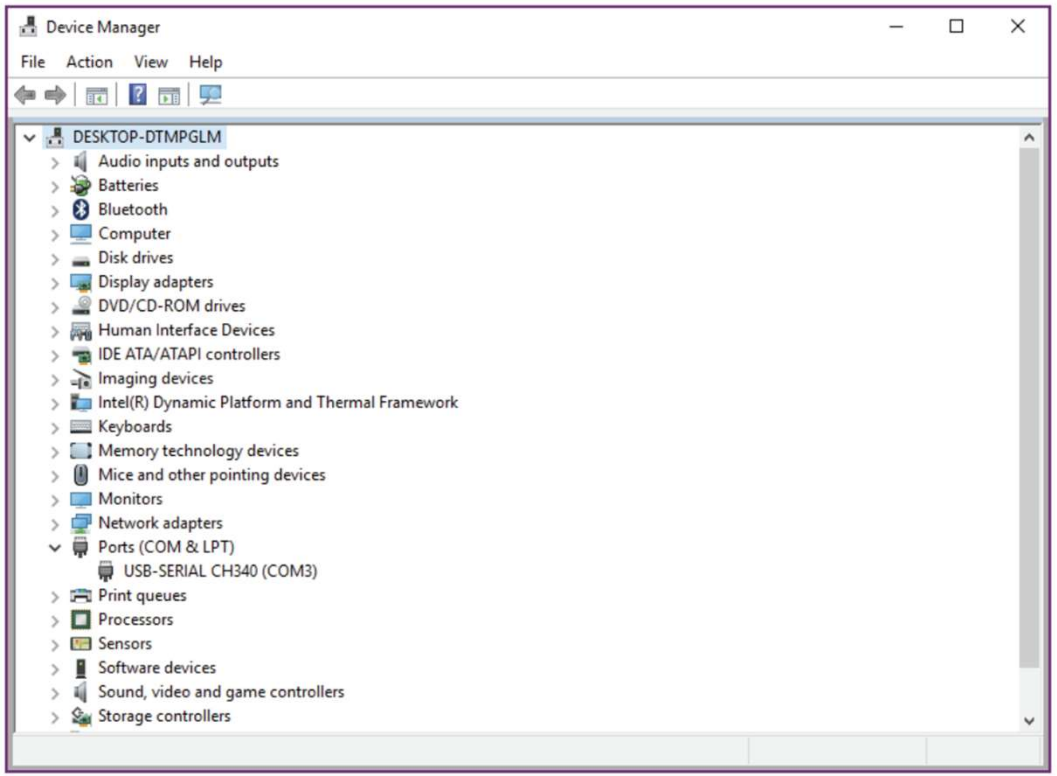


		
FEATURES	**Arduino Uno R3**	**Maker UNO**
Microcontroller	ATmega328P	ATmega328P
Programming IDE	Arduino IDE	Arduino IDE
Operating Voltage	5V	5V
Digital I/O Pins	20	20
PWM	6	6
Analog Input	6 (10-bit)	6 (10-bit)
UART	1	1
SPI	1	1
I2C	1	1
External Interrupt	2	2
Flash Memory	32 KB	32 KB
SRAM	2 KB	2 KB
EEPROM/Data Flash	1 KB	1 KB
Clock Speed	16 MHz	16 MHz
DC Current per I/O Pin	20 mA	20 mA
Power Supply	DC Adapter or USB	**USB only**
DC Current for 5V	1A	**USB Source**
DC Current for 3.3V	50 mA	**500 mA**
USB to Serial Chip	ATmega16u2	**CH340G**
Programmable LED	1 X at Pin 13	**12 X at digital Pin 2 to 13**
Programmable Push Button	No	**1 X at digital Pin 2**
Piezo Buzzer	No	**1 X at digital Pin 8**

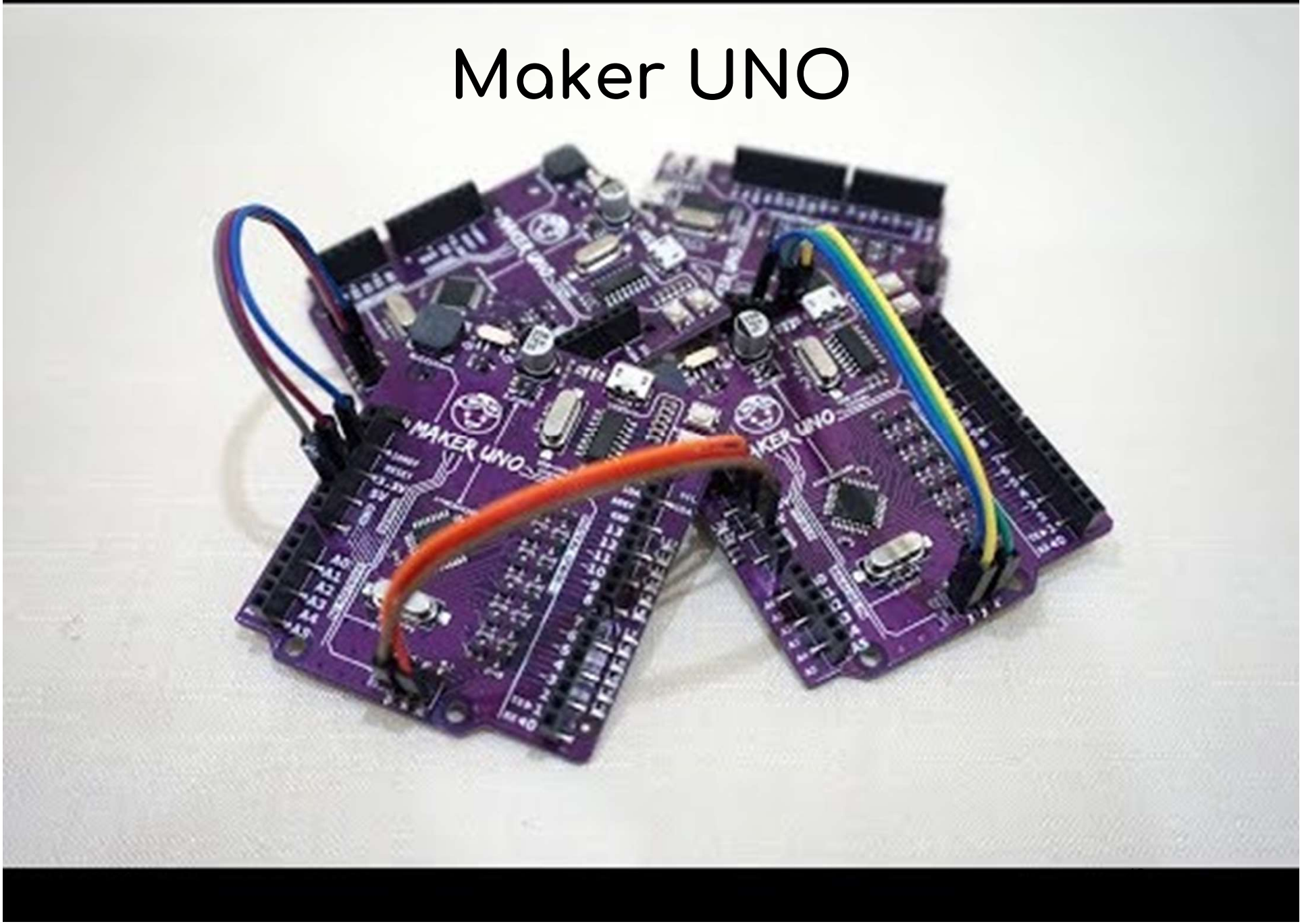
Install CH341 USB-Serial Driver

Download and install the USB CH341 Driver
Windows - https://cdn.cytron.io/makeruno/CH341SER.EXE
Mac OS - https://cdn.cytron.io/makeruno/CH341SER_MAC.ZIP

After installation is complete, your Maker UNO port should appears at Device Manager under Ports (COM & LPT) - e.g. USB-SERIAL CH340 (COM3). Please remember the port number.

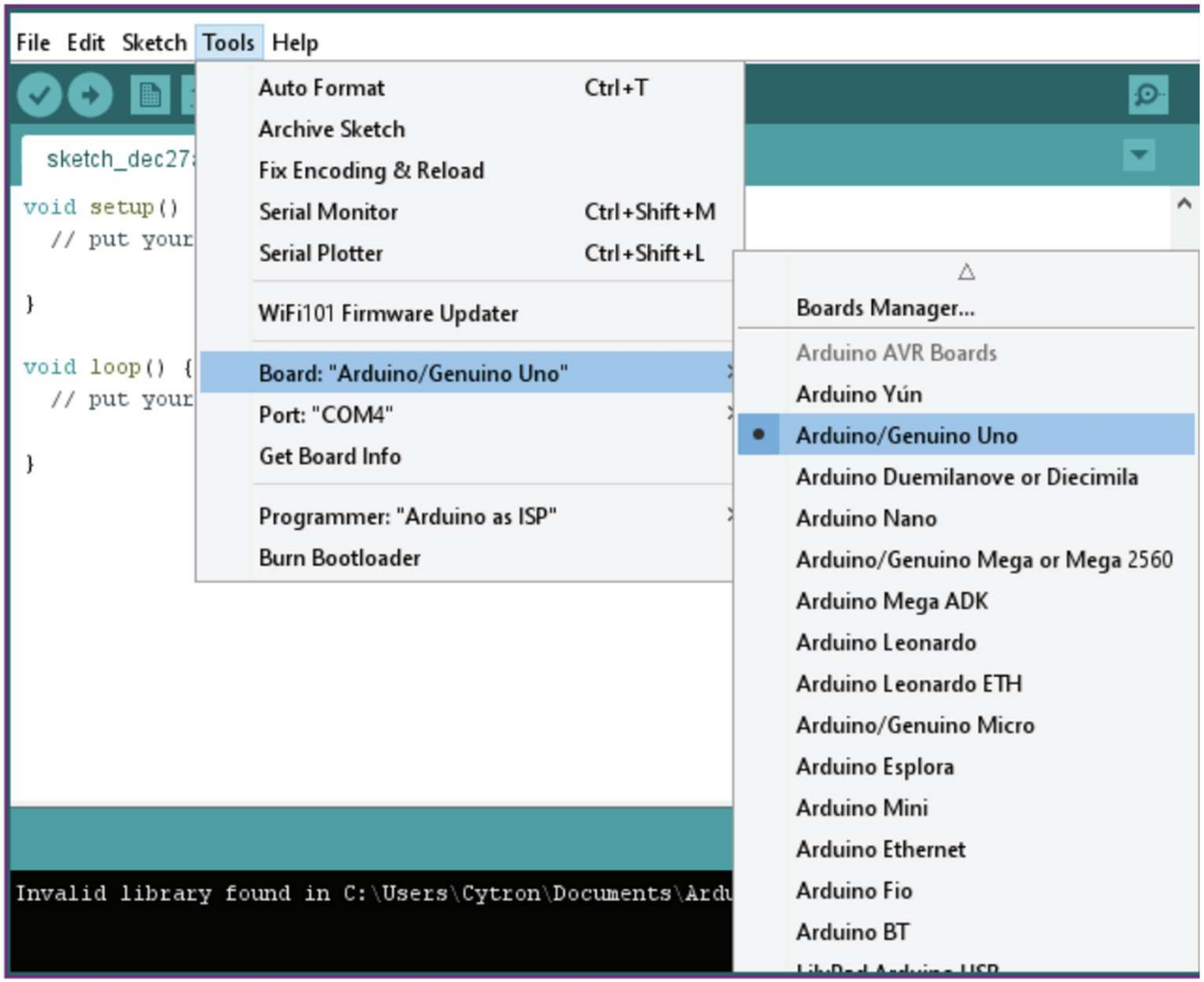


Maker UNO



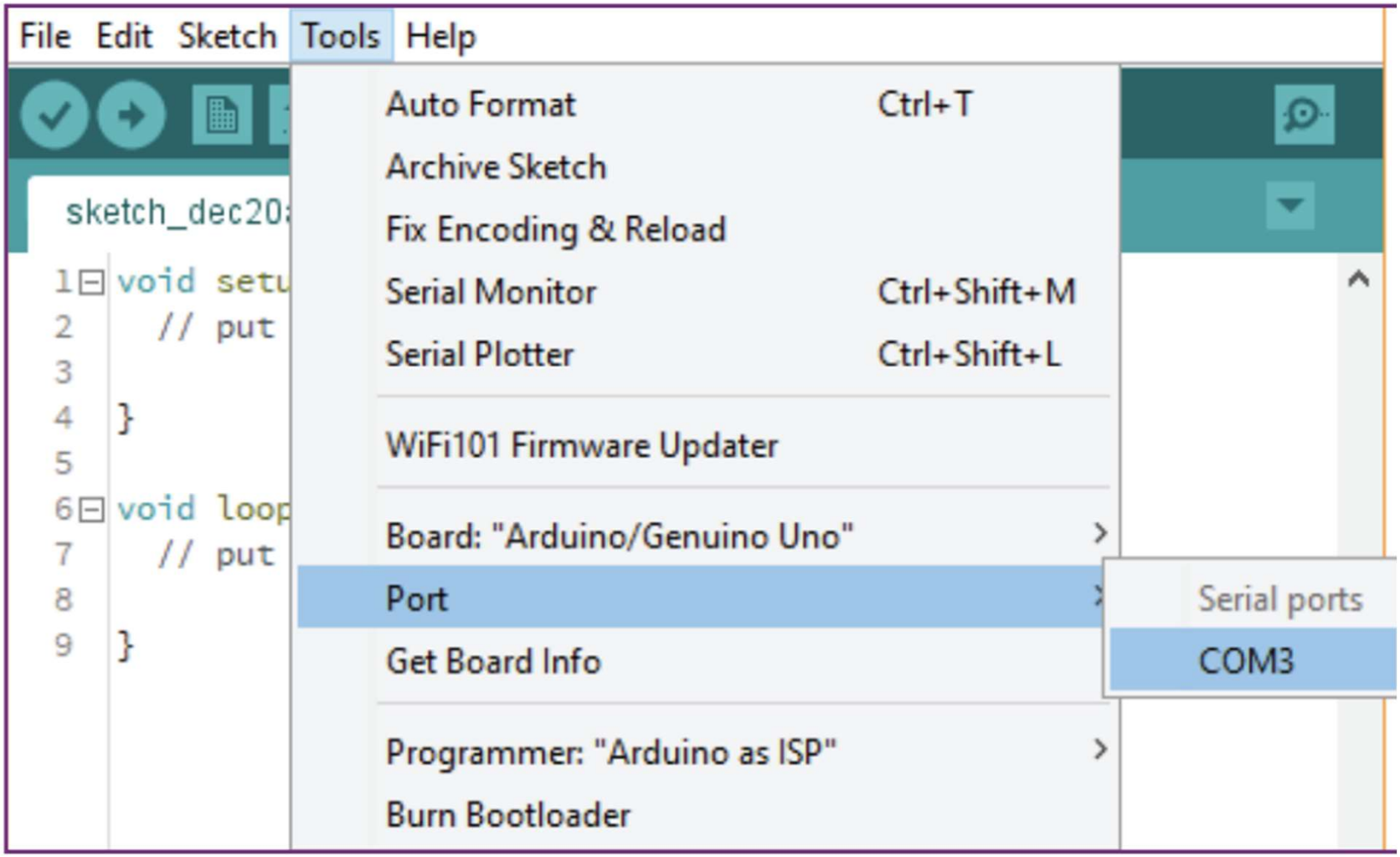
Select Board

- Select the Board : Tools->Board->Arduino Uno



Select Serial Port

- Select the Port: Tools->Port -> COM3 (Window)

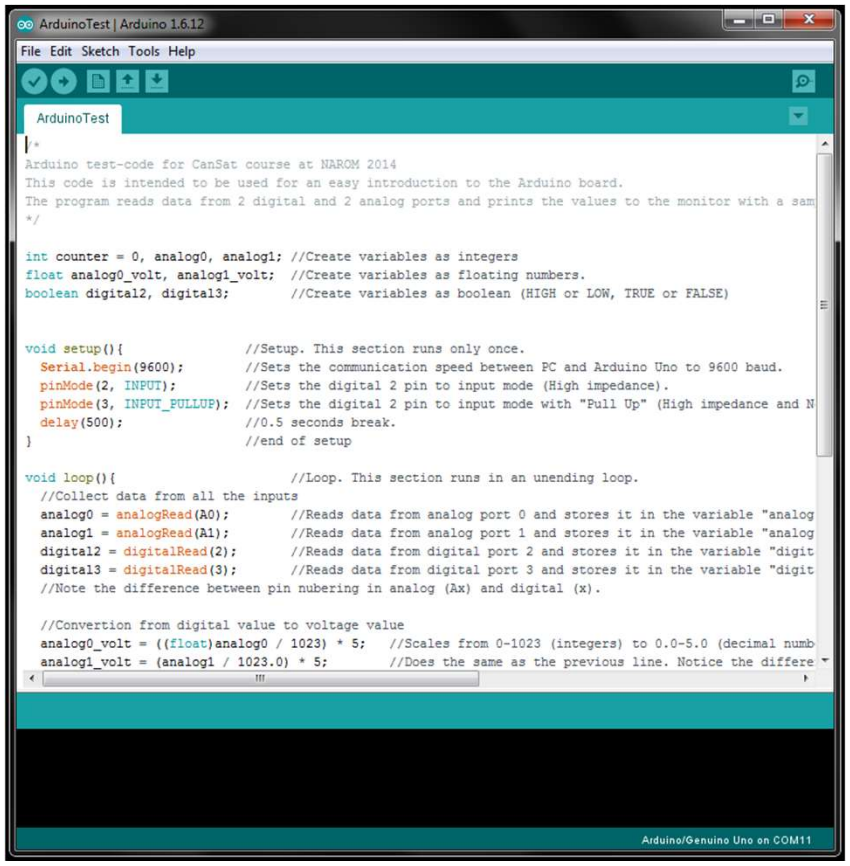


Arduino Communication Ports

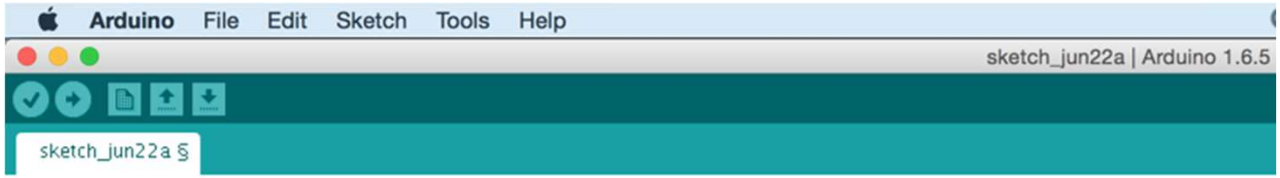
Platform	Port
Windows	Available in Device Manager
Mac	/dev/tty.usbmodem621 (or similar)
Linux	/dev/ttyACM0 (or similar)

Arduino Sketch

- A sketch is the name that Arduino uses for a program. It's the unit of code that is uploaded to and run on an Arduino board
- Everything between the /* and */, or after // is ignored by the Arduino when it runs the sketch



Arduino Program Structure



Define variables here

```
void setup() {
  // put your setup code here, to run once:
```

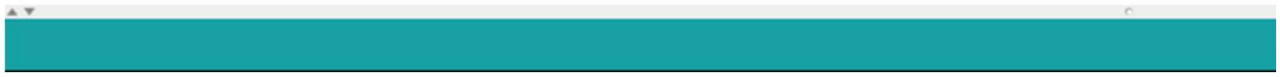
Define INPUT/OUTPUT Pins here

```
}

void loop() {
  // put your main code here, to run repeatedly:
```

Create your main program here

```
}
```

Sketch = Program = Source Code



Variable

- A variable is a place for storing a piece of data. It has a name, a type, and a value.
- For example, the line from the Blink sketch above declares a variable with the name ledPin, the type int, and an initial value of 13.
- It's being used to indicate which Arduino pin the LED is connected to. Every time the name ledPin appears in the code, its value will be retrieved.

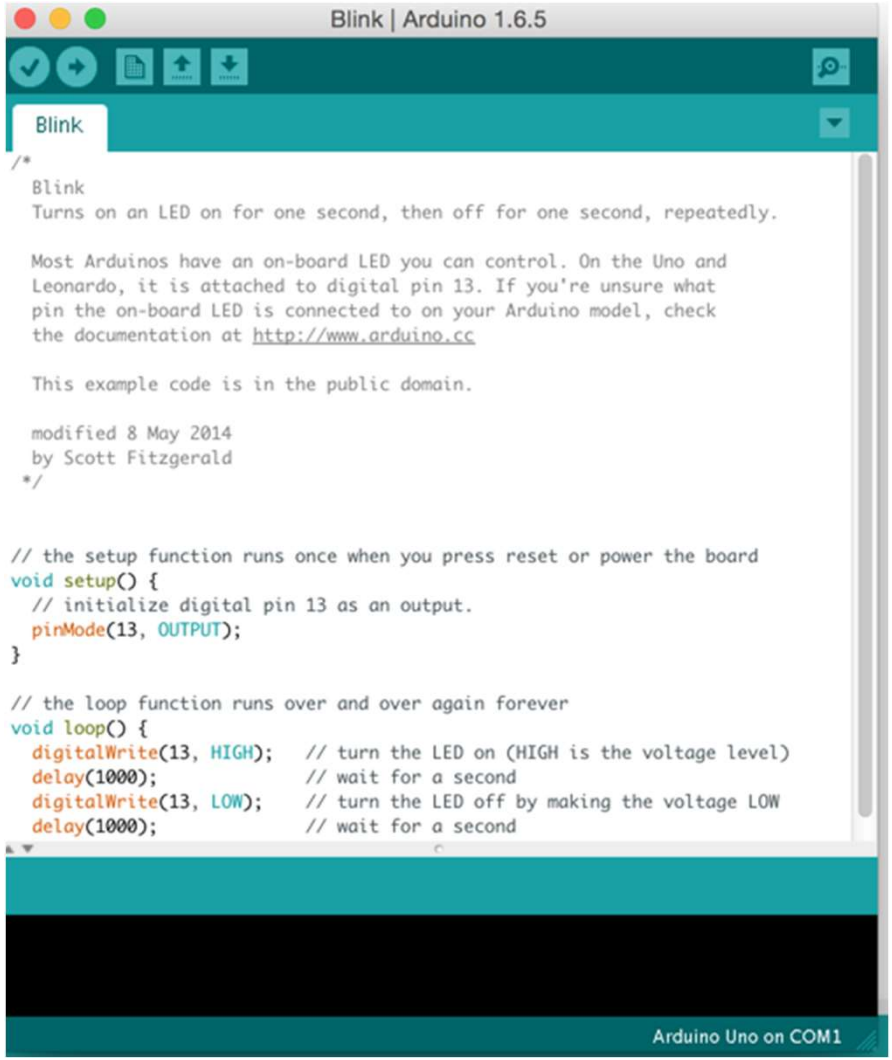
```
int ledPin = 13;
```

Function

- A function (otherwise known as a procedure or sub-routine) is a named piece of code that can be used from elsewhere in a sketch.
- There are two special functions that are a part of every Arduino sketch: setup() and loop().
- The setup() is called once, when the sketch starts. It's a good place to do setup tasks like setting pin modes or initializing libraries.
- The loop() function is called over and over and is heart of most sketches. You need to include both functions in your sketch, even if you don't need them for anything.

```
void setup()
{

  pinMode(ledPin, OUTPUT);
}
```

Arduino Sketch Demo



- Open File->Example-> 0.1Basic->Blink
- Upload/Flash the sketch to the Arduino board
- Observe the LED blink

Blink Sketch

```
// the setup function runs once when you press reset
or power the board

void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(LED_BUILTIN, HIGH);   // turn the LED on
  delay(1000);                       // wait for a
second
  digitalWrite(LED_BUILTIN, LOW);    // turn the LED off
  delay(1000);                       // wait for a
second
}
```

pinMode(), digitalWrite(), and delay()

- The pinMode() function configures a pin as either an input or an output.
- To use it, you pass it the number of the pin to configure and the constant INPUT or OUTPUT.
- When configured as an input, a pin can detect the state of a sensor like a pushbutton; this is discussed in a later tutorial.
- As an output, it can drive an actuator like an LED.

  digitalWrite(ledPin, HIGH);

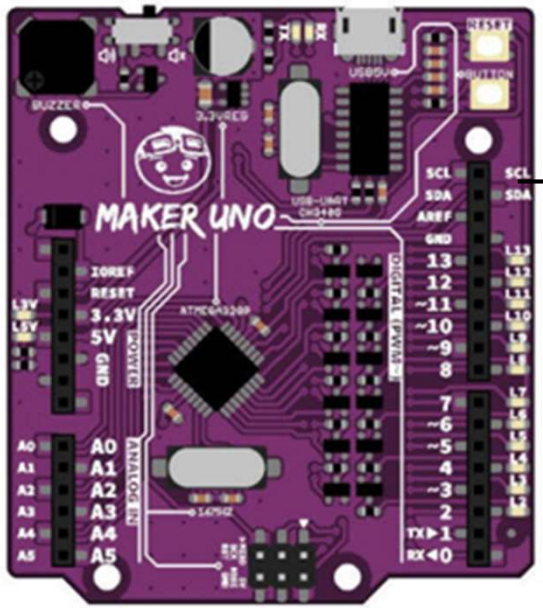
- The delay() causes the Arduino to wait for the specified number of milliseconds before continuing on to the next line. There are 1000 milliseconds in a second, so the line:

Activity: Arduino Sketch

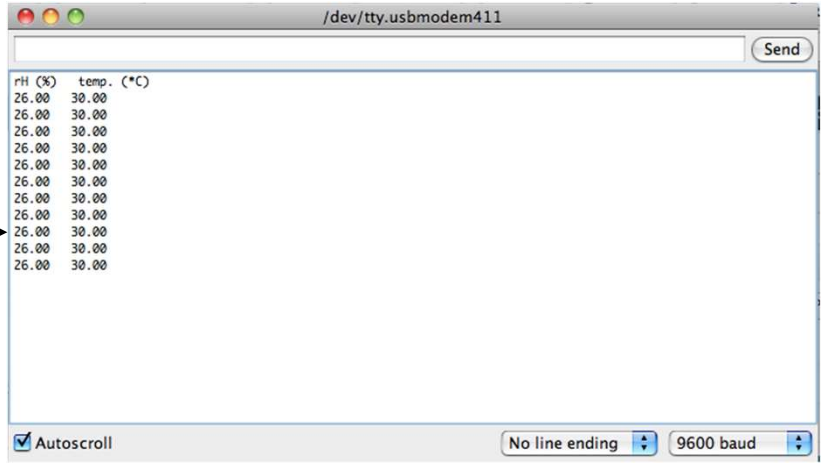
- Change the code so that the LED is on for 100 milliseconds and off for 1000.
- Change the code so that the LED turns on when the sketch starts and stays on.

Serial Communication

- Used for communication between the Arduino board and a computer or other devices.
- All Arduino boards have at least one serial port (also known as a UART or USART), and some have several.

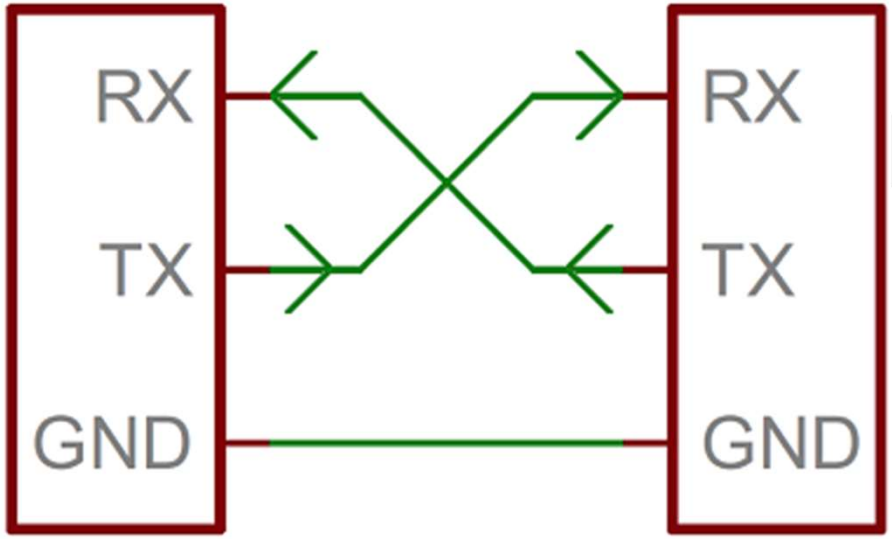


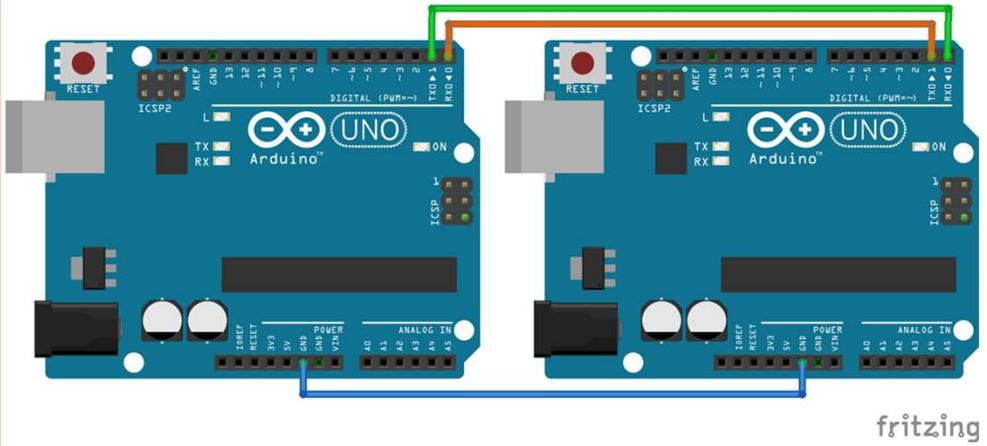
COM Port



RX/TX Serial Bus

- A serial bus consists of just two wires - the transmitter TX wire for sending data and receiver RX wire for receiving data.





Serial Communication Commands

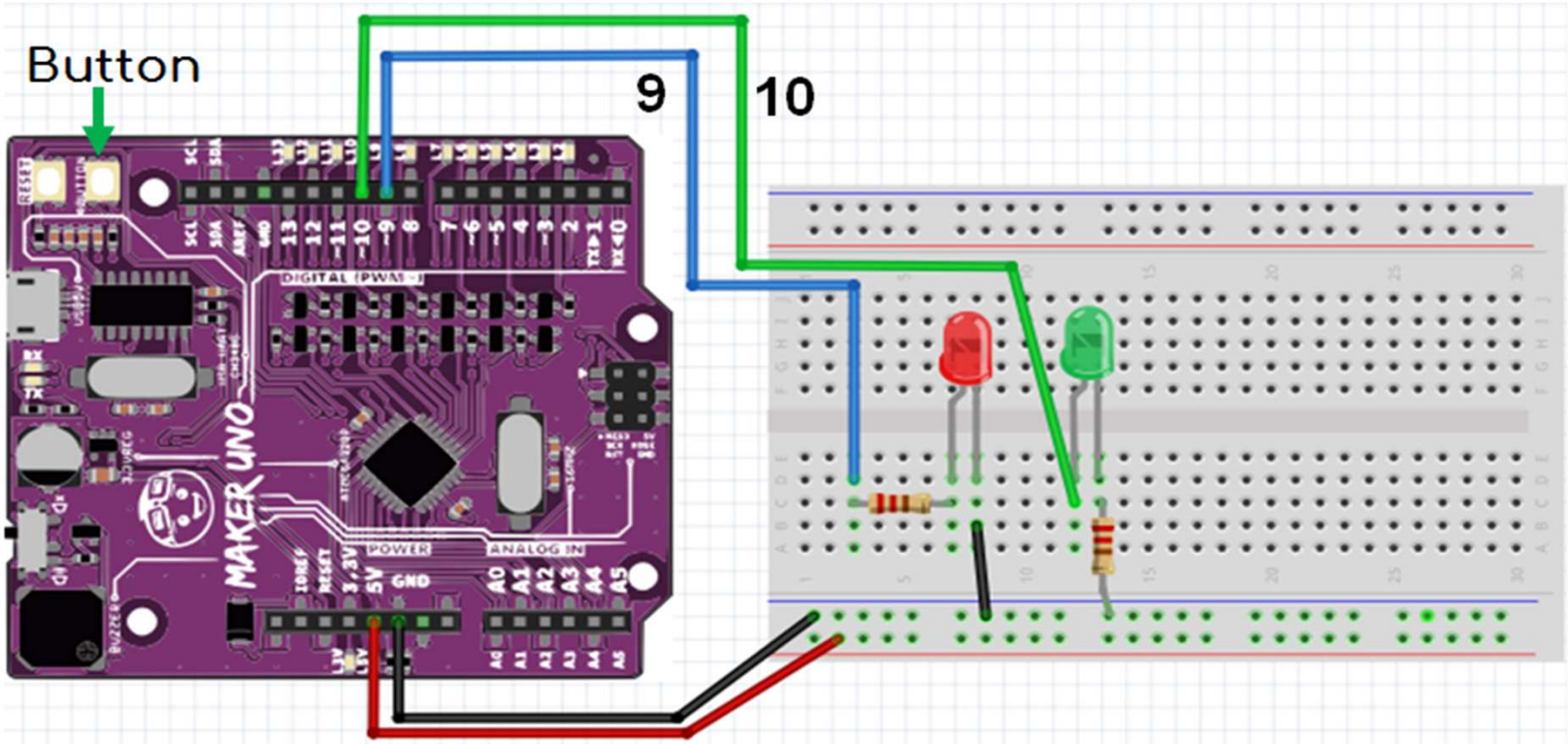
- Serial.begin() - Sets the data rate in bits per second (baud) for serial data transmission.
- Serial.end() - Disables serial communication, allowing the RX and TX pins to be used for general input and output
- Serial.print() - Prints data to the serial port as human-readable ASCII text.
- Serial.println() - Prints data to the serial port as human-readable ASCII text followed by a carriage return character (ASCII 13, or '\r') and a newline character (ASCII 10, or '\n')
- Serial.available() - Get the number of bytes (characters) available for reading from the serial port.

Serial Communication Example

```
void setup() {
    Serial.begin(9600);
    Serial.println("Hello World");
}

void loop()
     if (Serial.available () > 0)
    {
        val = Serial.read ();
         }
}
```

Activity: Serial Communication

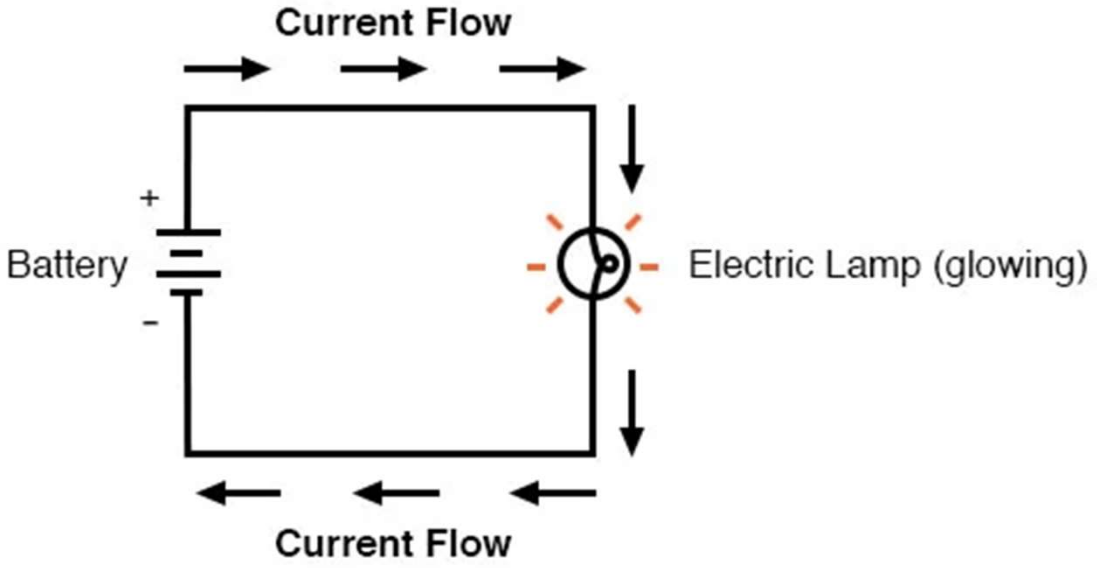
- Code the sketch to turn ON and OFF the LED using the keyboard.
  - 1 - Turn On
  - 2 - Turn Off



Topic 2
Introduction to Basic Electronics

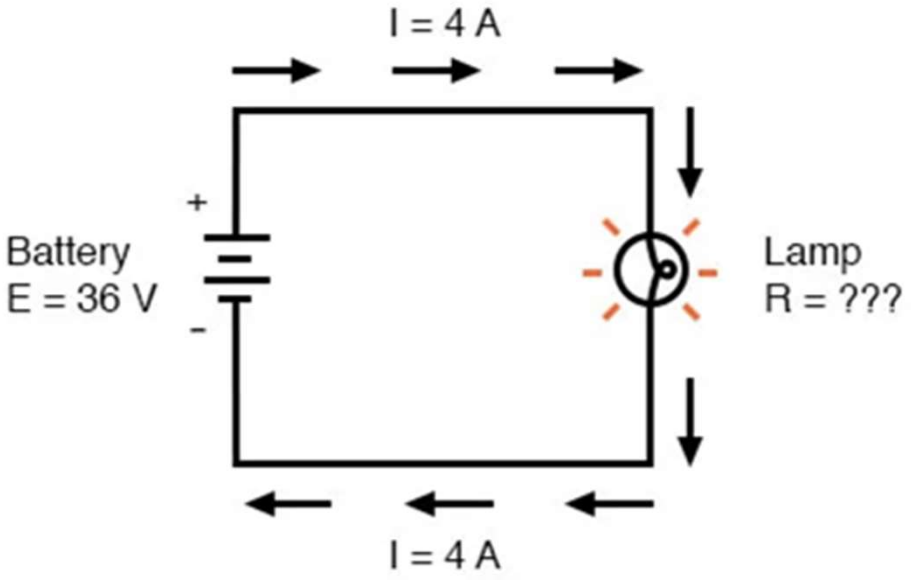
Electric Circuit

- An electric circuit is formed when a conductive path is created to allow electric charge to continuously move.
- This continuous movement of electric charge through the conductors of a circuit is called a current, and it is often referred to in terms of "flow," just like the flow of a liquid through a hollow pipe.



Ohm's Law

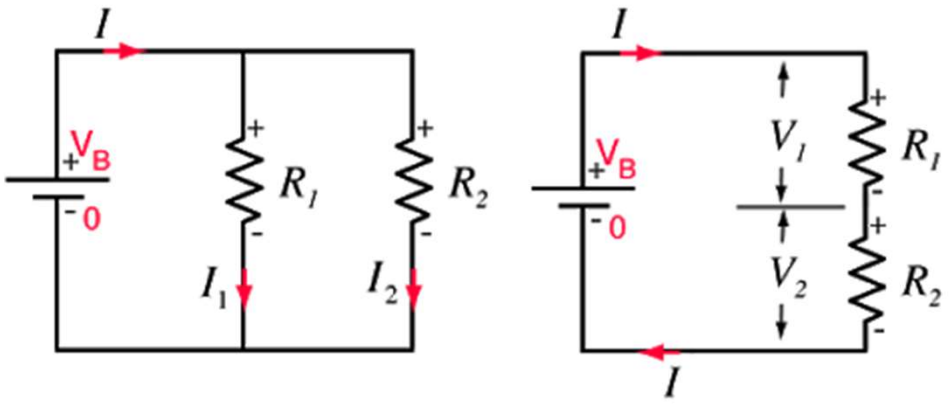
- Ohm's Law is a very simple and useful tool for analyzing electric circuits.
- It is used so often in the study of electricity and electronics that it needs to be committed to memory by the serious student.



$$R = \frac{E}{I} = \frac{36\ V}{4\ A} = 9\ \Omega$$

Series Vs. Parallel Circuits

- There are two different ways in which you can wire things together called series and parallel.
- When things are wired in series, things are wired one after another, such that electricity has to pass through one thing, then the next thing, then the next, and so on.
- When things are wired in parallel, they are wired side by side, such that electricity passes through all of them at the same time, from one common point to another common poin



Parallel resistors

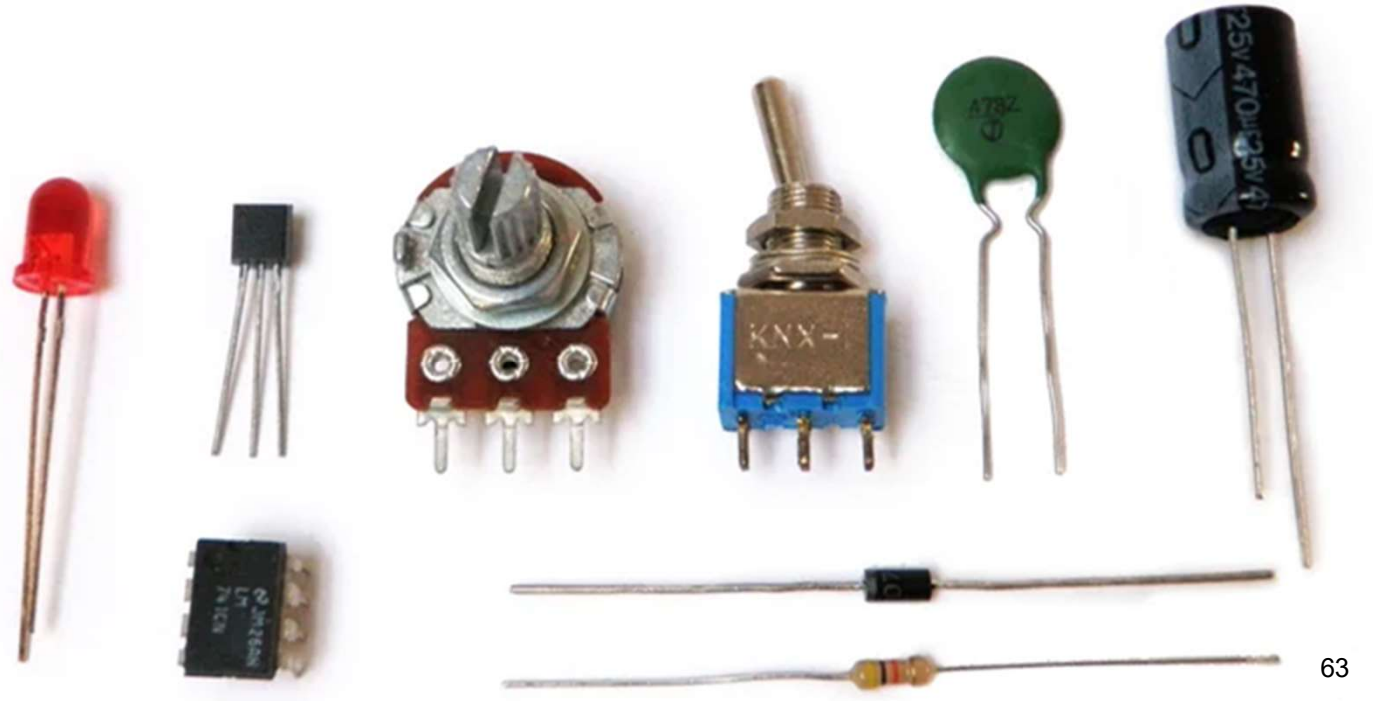
$$\frac{1}{R_{equivalent}} = \frac{1}{R_1} + \frac{1}{R_2}$$

Series resistors

$$R_{equivalent} = R_1 + R_2$$

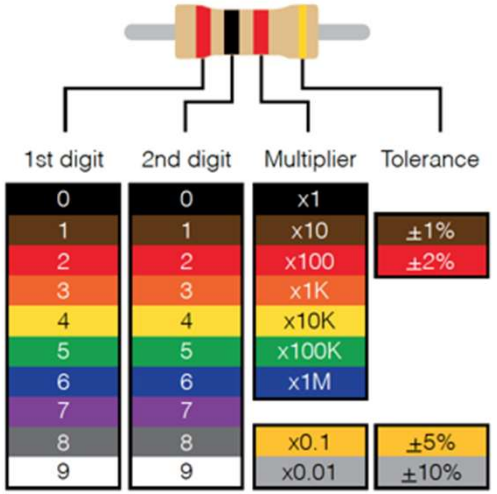
Basic Components

- In order to build circuits, you will need to become familiar with a few basic components.
- These components may seem simple, but are the bread and butter of most electronics projects.
- Thus, by learning about these few basic parts, you will be able to go a long way.



Resistor

- Resistors add resistance to the circuit and reduces the flow of electrical current. It is represented in a circuit diagram as a pointy squiggle with a value next to it.
- The different markings on the resistor represent different values of resistance. These values are measured in ohms. You can use the online tool http://www.dannyg.com/examples/res2/resistor.htm
- Resistors also come with different wattage ratings. For most low-voltage DC circuits, 1/4 watt resistors should be suitable.

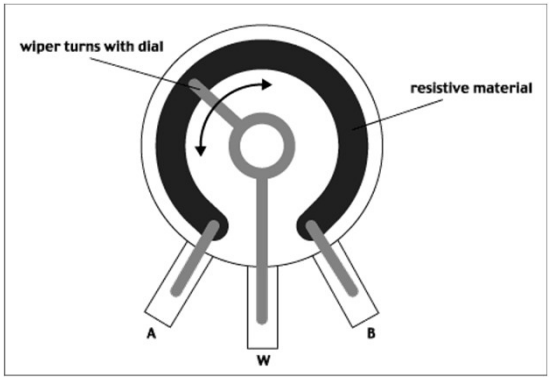


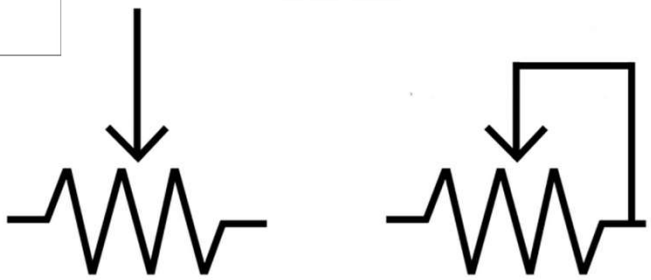
2, 2, x 10 = 220Ω

1, 1, x 1,000 = 11KΩ

Potentiometer / Variable Resistor

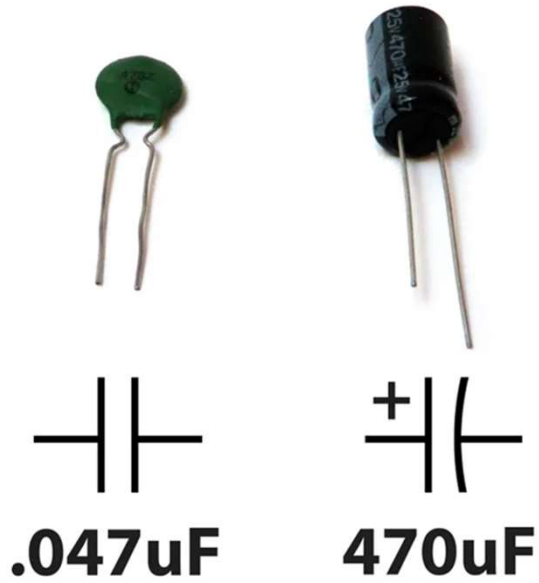
- Potentiometers are variable resistors. In plain English, they have some sort of knob or slider that you turn or push to change resistance in a circuit.
- If you have ever used a volume knob on a stereo or a sliding light dimmer, then you have used a potentiometer.





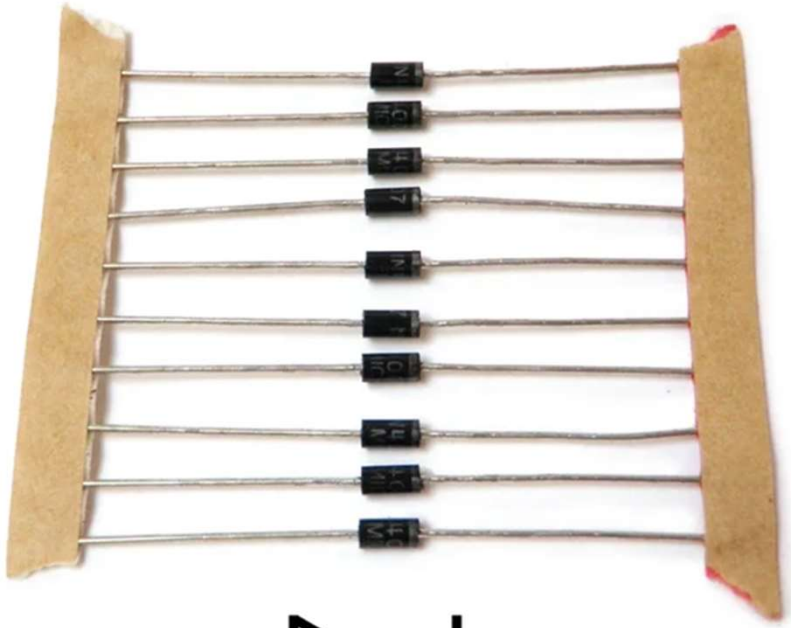
Capacitors

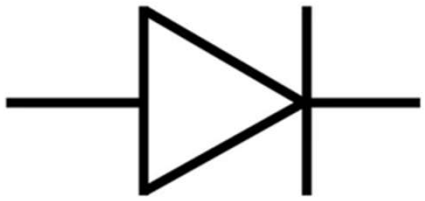
- A capacitor is a component that stores electricity and then discharges it into the circuit when there is a drop in electricity. You can think of it as a water storage tank that releases water when there is a drought to ensure a steady stream.
- Capacitors are measured in Farads. The values that you will typically encounter in most capacitors are measured in picofarad (pF), nanofarad (nF), and microfarad (uF)
- The most commonly encountered types of capacitors are ceramic disc capacitors that look like tiny M&Ms with two wires sticking out of them and electrolytic capacitors that look more like small cylindrical tubes with two wires coming out the bottom



Diodes

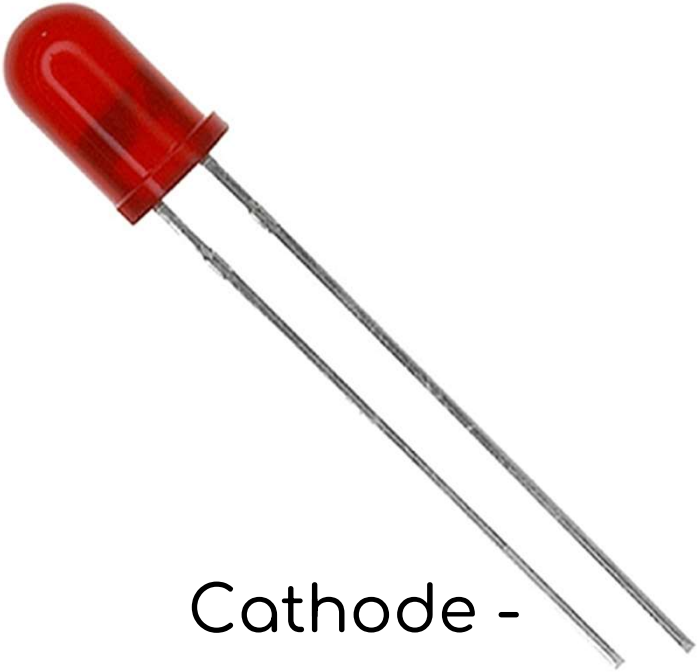
- Diodes are components which are polarized. They only allow electrical current to pass through them in one direction. This is useful in that it can be placed in a circuit to prevent electricity from flowing in the wrong direction.
- Another thing to keep in mind is that it requires energy to pass through a diode and this results in a drop of voltage. This is typically a loss of about 0.7V.

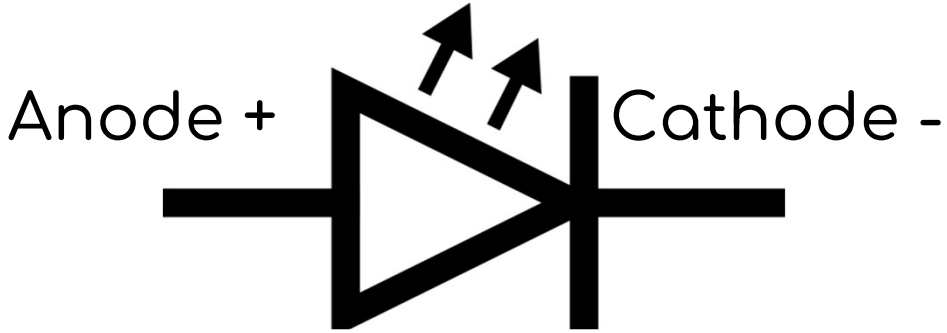




Light Emitting Diode (LED)

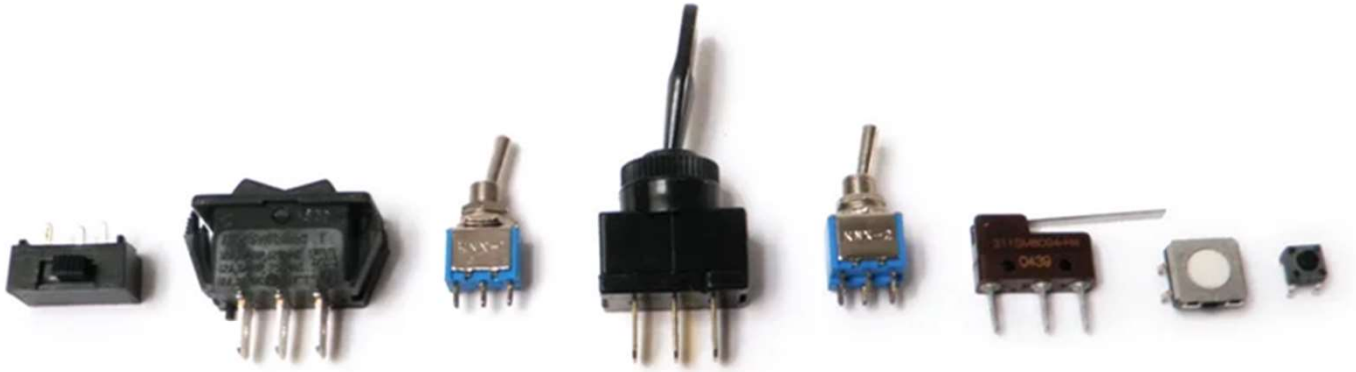
- Light-emitting diode (LED) is a semiconductor device that emits light when an electric current is passed through it.
- Light is produced when the particles that carry the current (known as electrons and holes) combine together within the semiconductor material.





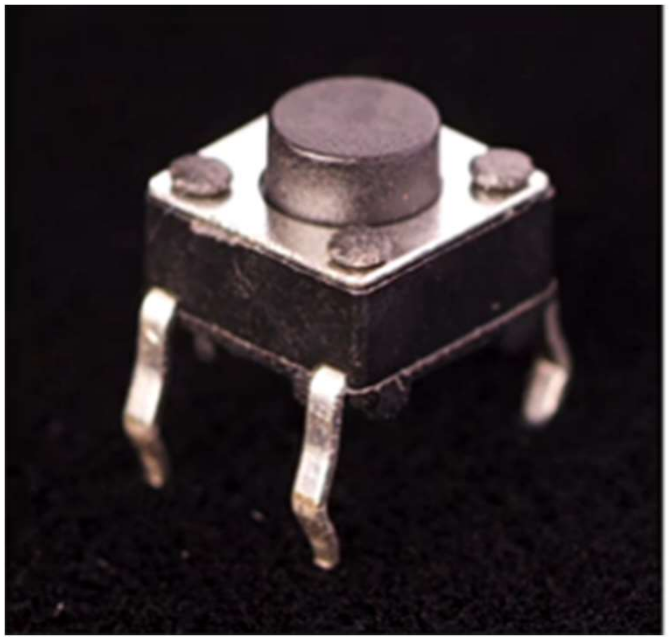
Switches

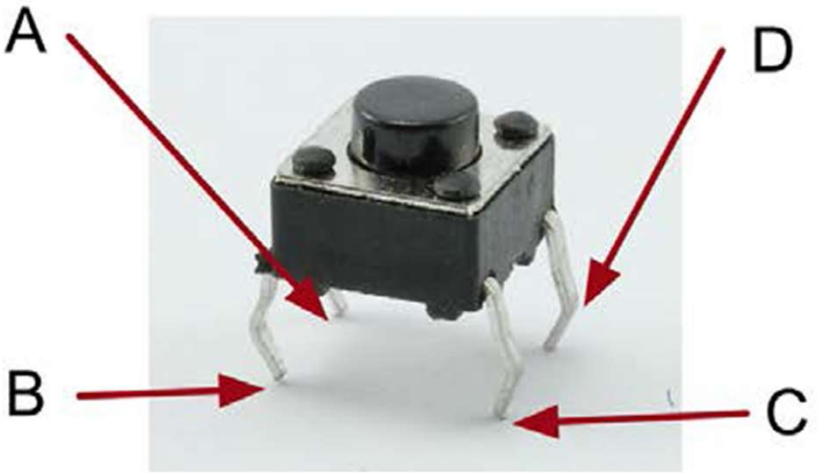
- A switch is basically a mechanical device that creates a break in a circuit. When you activate the switch, it opens or closes the circuit. This is dependent on the type of switch it is.
- Normally open (N.O.) switches close the circuit when activated.
- Normally closed (N.C.) switches open the circuit when activated.

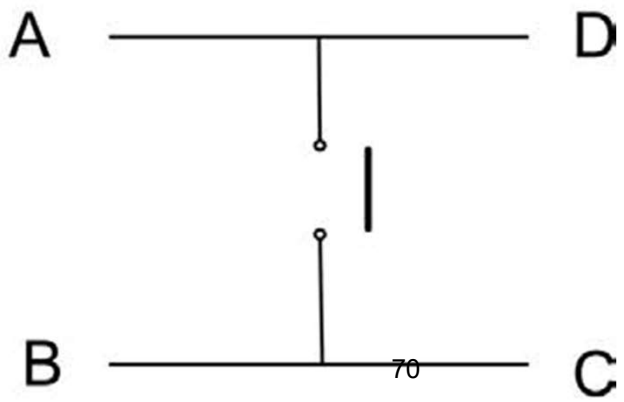


Push Button Switch

- Press to Turn On
- Release to Turn Off
- Generate Digital Input

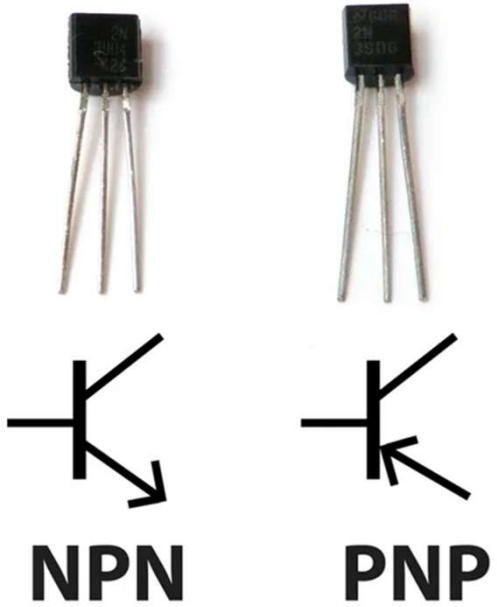






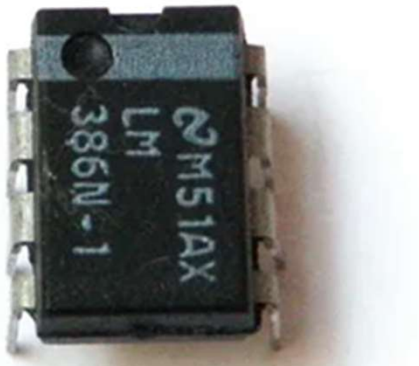
Transistors

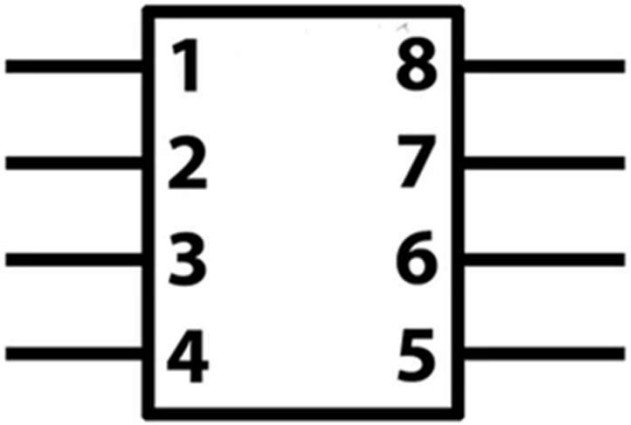
- A transistor takes in a small electrical current at its base pin and amplifies it such that a much larger current can pass between its collector and emitter pins. The amount of current that passes between these two pins is proportional to the voltage being applied at the base pin.
- There are two basic types of transistors, which are NPN and PNP. These transistors have opposite polarity between collector and emitter.



Integrated Circuits

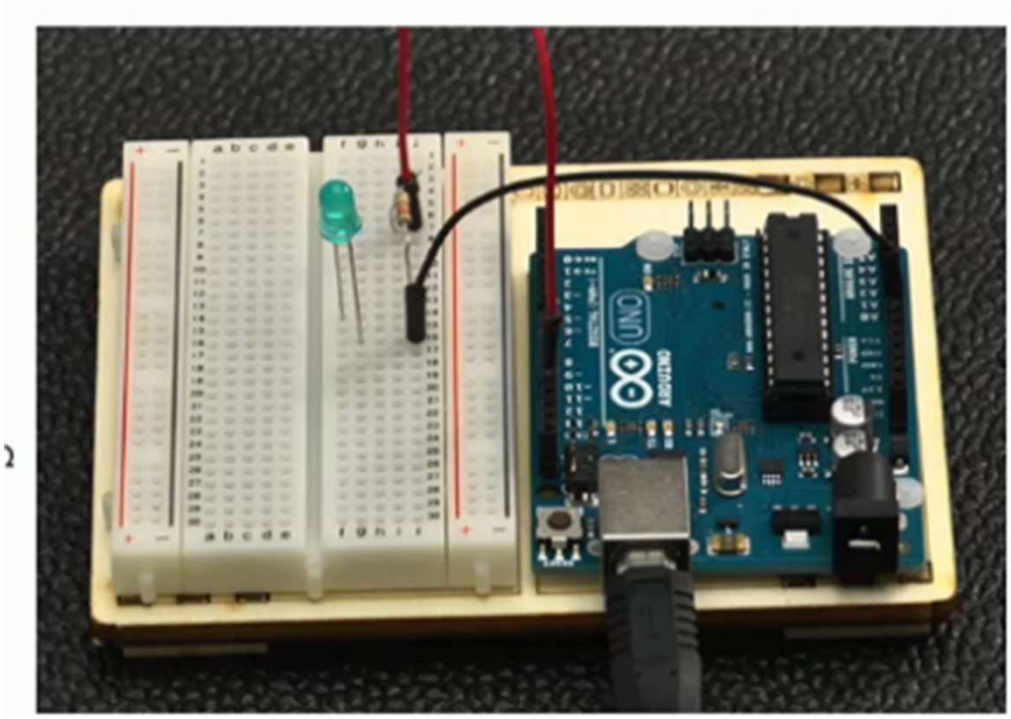
- An integrated circuit is an entire specialized circuit that has been miniaturized and fit onto one small chip with each leg of the chip connecting to a point within the circuit.
- These miniaturized circuits typically consist of components such as transistors, resistors, and diodes.

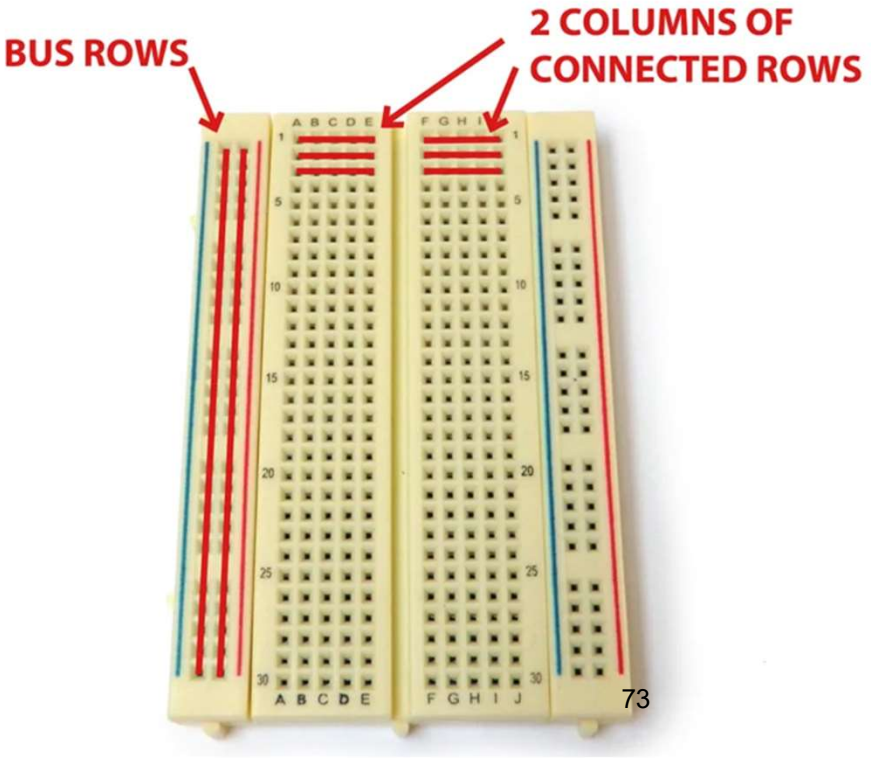




Solderless Breadboard

- Breadboards are special boards for prototyping electronics. They are covered with a grid of holes, which are split into electrically continuous rows.
- A solderless breadboard allow you to prototype an electronics circuit without soldering



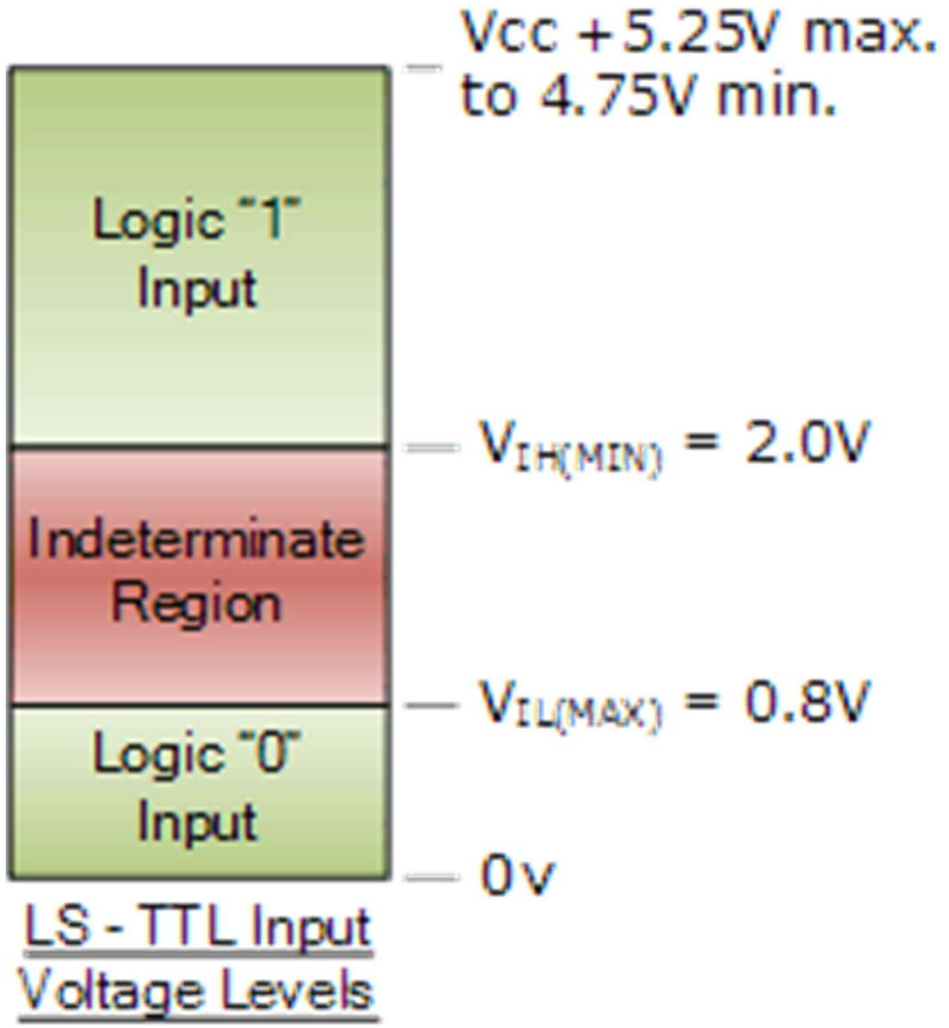


Digital I/O

- Digital I/O stands for Digital Input and Output. Digital Inputs allow a microcontroller to detect logic states, and Digital Outputs allow a microcontroller to output logic states.
- Each digital I/O can be individually configured to one of 3 states: input, output-high, or output-low.
- Digital Input: A digital input detects if a voltage is above/below a specific threshold. If the voltage is higher than some value, the computer will detect the digital input as high/1. If the voltage is lower than some value, the computer will detect the digital input as low/0.
- Digital Output: A digital output allows you to control a voltage with a computer. If the computer instructs the output to be high, the output will produce a voltage (generally about 5 or 3.3 volts). If the computer instructs the output to be low, it is connected to ground and produces no voltage.

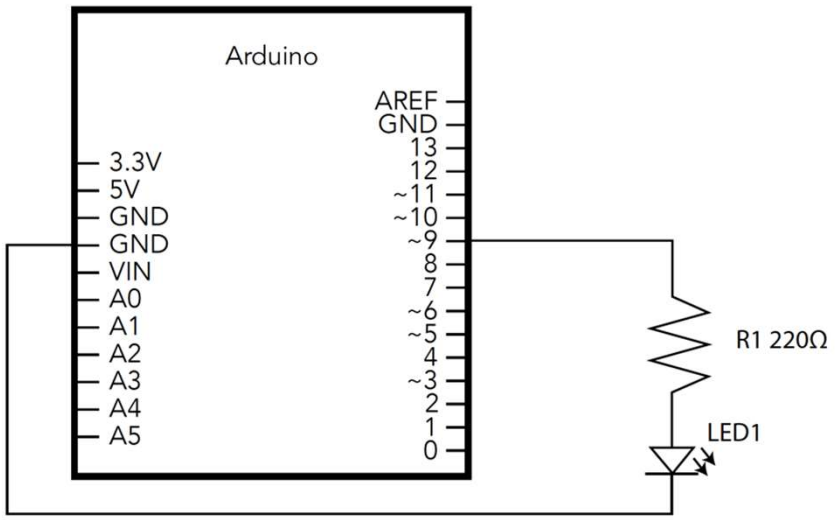
Digital Signal Logic

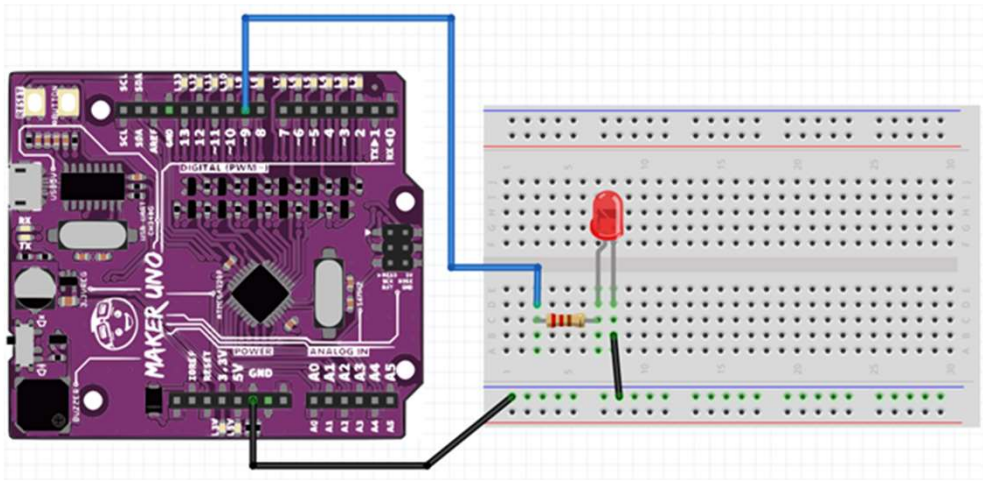
- 3 logic states
  - HIGH (Logic 1)
  - LOW (Logic 0)
  - Floating
    (can be logic 0 or 1)

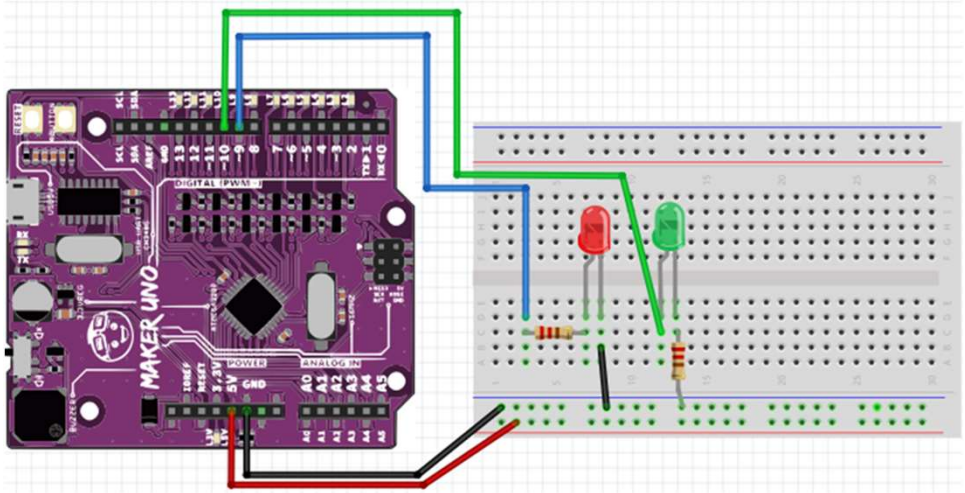


Activity: Blinking LED

- Write a sketch to turn on red LED
- Write a sketch to blink red LED
- Write a sketch to alternate blinking 2 LEDs

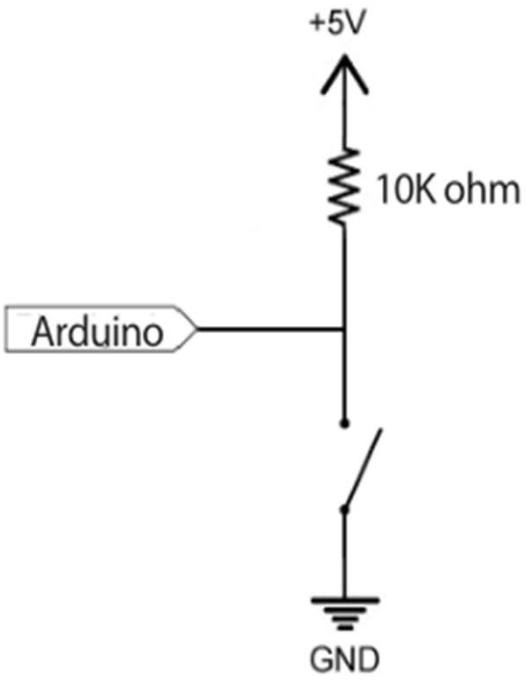






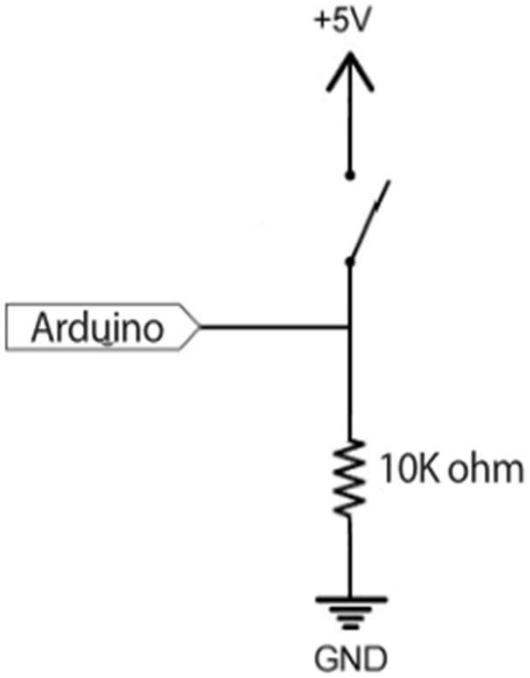
Pull up and Pull down Resistor

Pull-up resistor



It is HIGH when the switch is not pressed

Pull-down resistor



It is LOW when the switch is not pressed

digitalRead

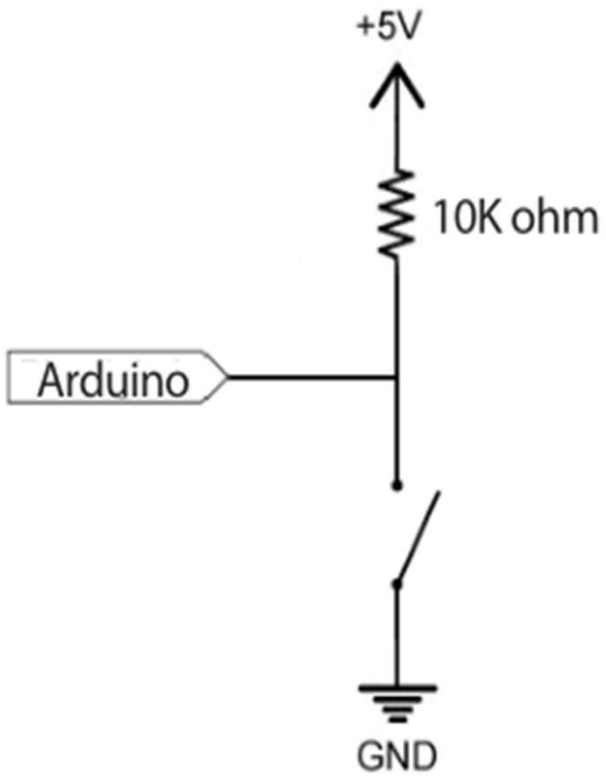
- Reads the value from a specifiec digital pin, either HIGH or LOW

- Syntax:

digitalRead(PIN, INPUT_PULLUP);

- For Example:

digitalRead(2,INPUT_PULLUP);

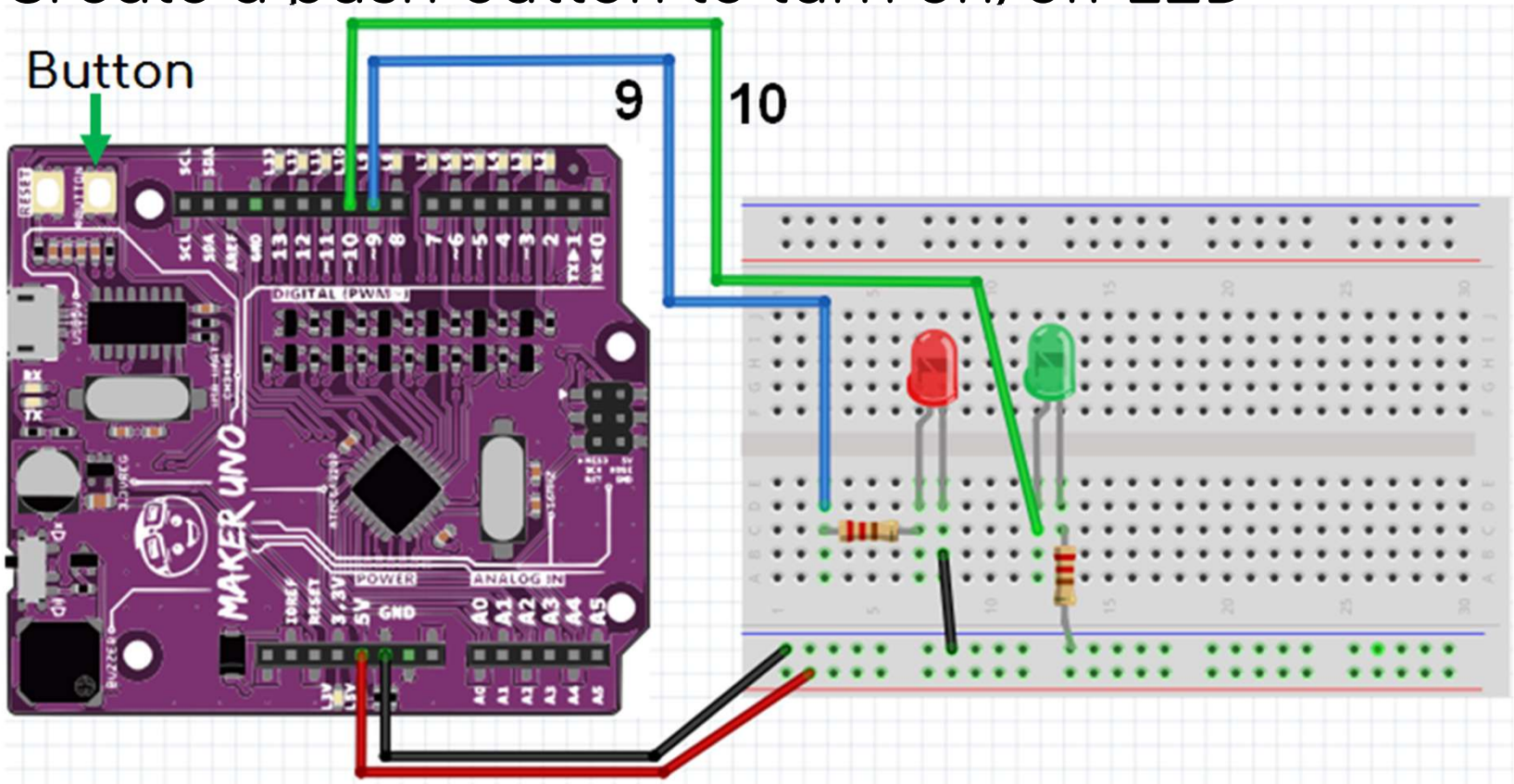


digitalRead Example

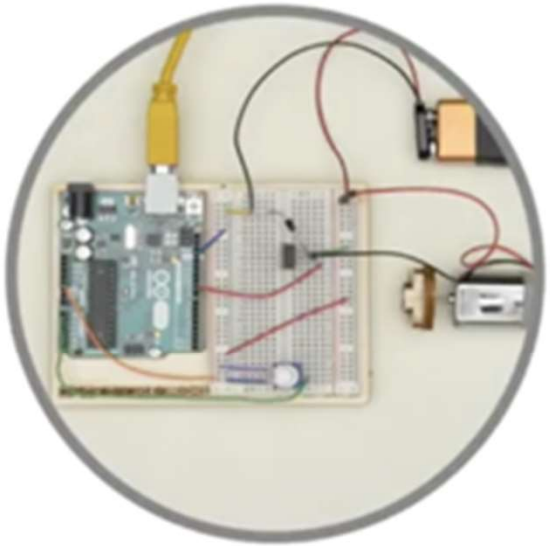
```
int ledPin = 13;  // LED connected to digital pin 13
int inPin = 7;    // pushbutton connected to digital pin 7
int val = 0;      // variable to store the read value

void setup() {
  pinMode(ledPin, OUTPUT);
  pinMode(inPin, INPUT);
}
void loop() {
  val = digitalRead(inPin);   // read the input pin
  digitalWrite(ledPin, val);  // sets the LED to the button's
value
}
```

Activity: Push Button

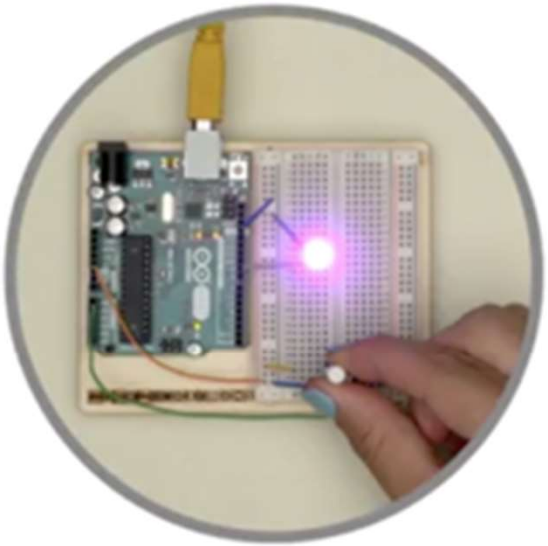
- Push button act as a digital input device. Maker UNO is able to sense 2 states for digital input, i.e. HIGH and LOW. Push the button and the LED will turn ON!
- Create a push button to turn on/off LED



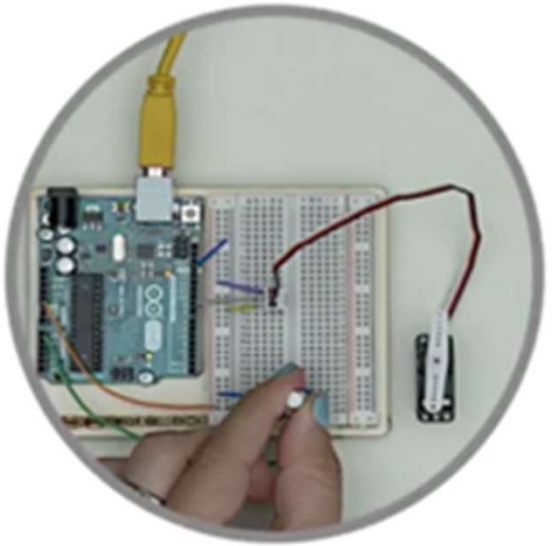
Analog Output Applications



Motor Speed



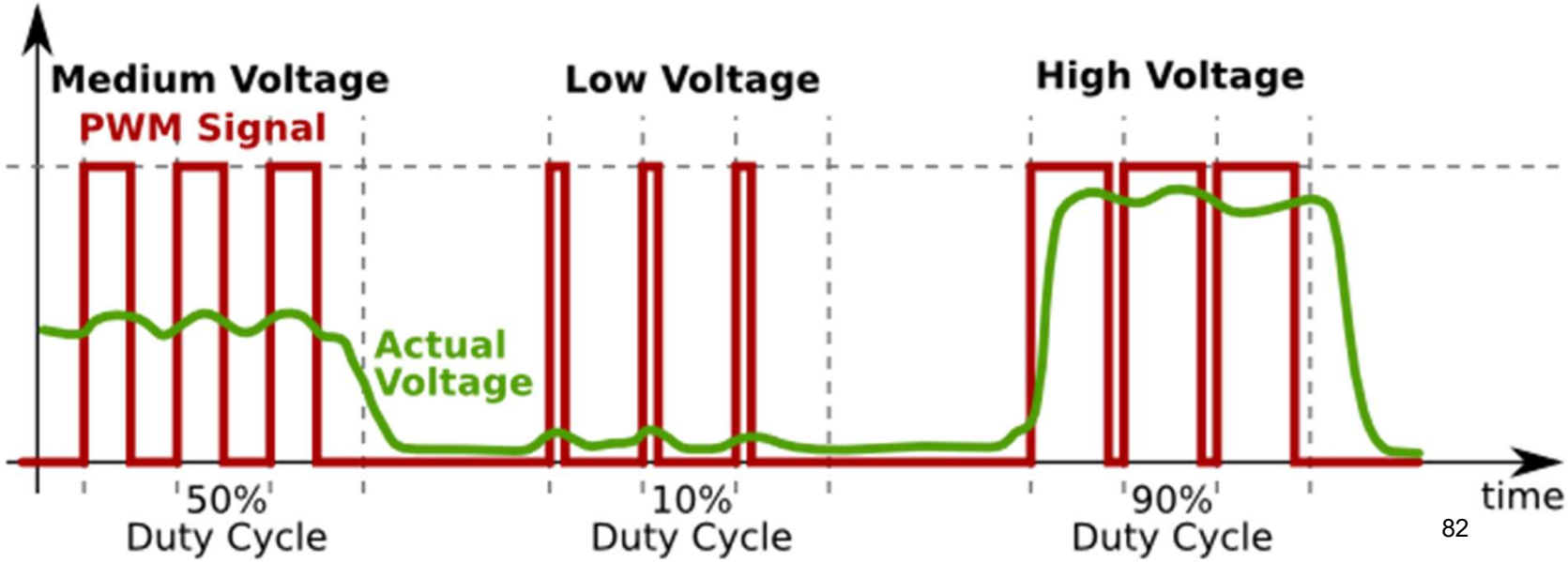
LED Brightness



Servo Angle

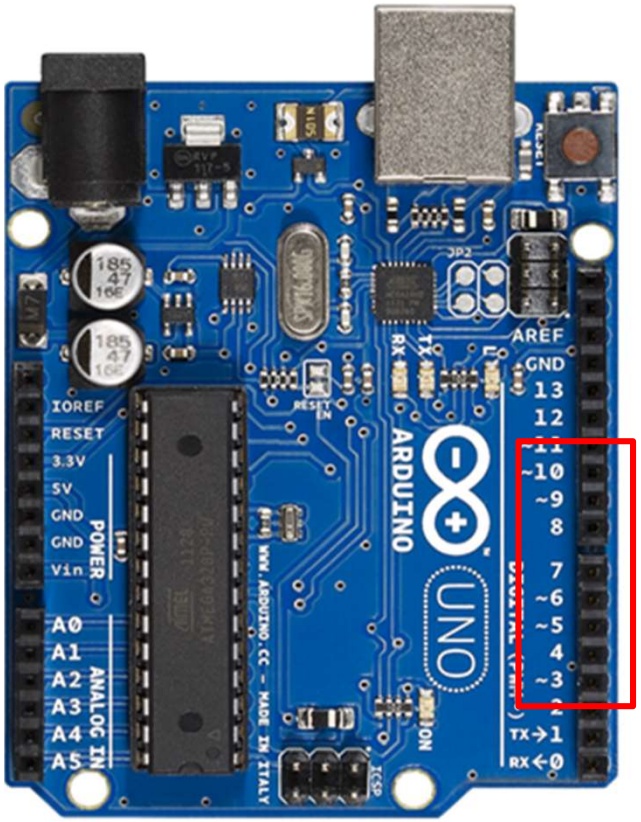
Pulse Width Modulation (PWM)

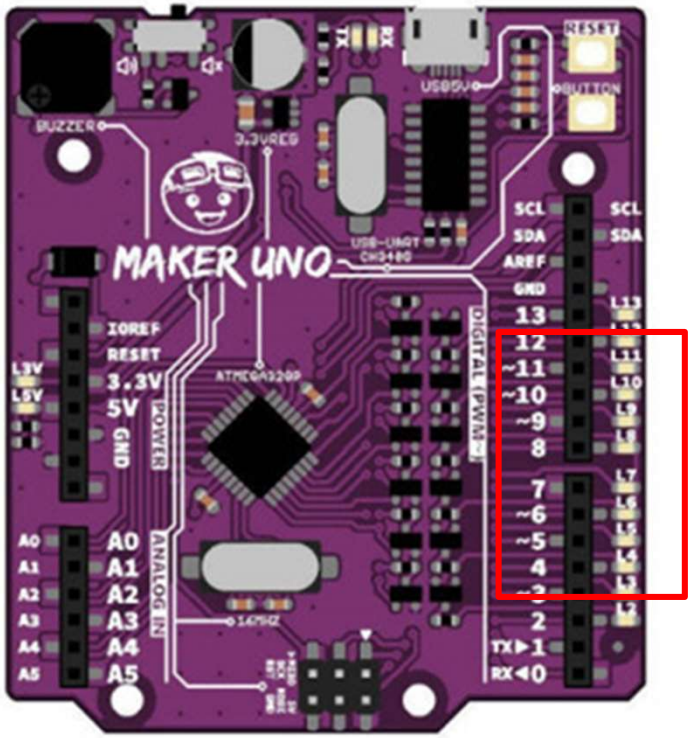
- Pulse Width Modulation, or PWM, is a technique for getting analog results with digital means. Pulse width modulation is used in a variety of analog applications
- Digital control is used to create a square wave, a signal switched between on and off. By varying the pulse width or duty cycle, we create an analog voltage.



PWM Pins

- A few PINs on the Arduino Digital PINs allow to modify the output to mimic analog signal
- Pin 3, 5, 6, 9, 10, 11
- They are indicated by a "~" besides the pin no.





analogWrite

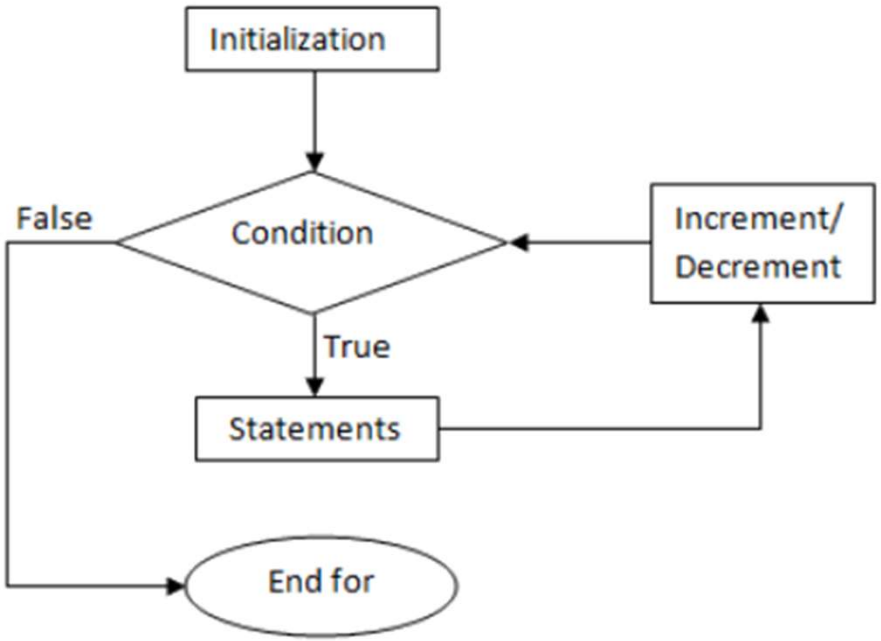
- Writes an analog value (PWM wave) to a pin. Can be used to light a LED at varying brightnesses or drive a motor at various speeds. After a call to analogWrite(), the pin will generate a steady rectangular wave of the specified duty cycle until the next call to analogWrite().

- Syntax
- analogWrite(PIN, value)

- Example:
- analogWrite(9, 125)

Activity: Fade an LED

- Write a simple sketch to fade the LED brightness from low to high to low repeatedly



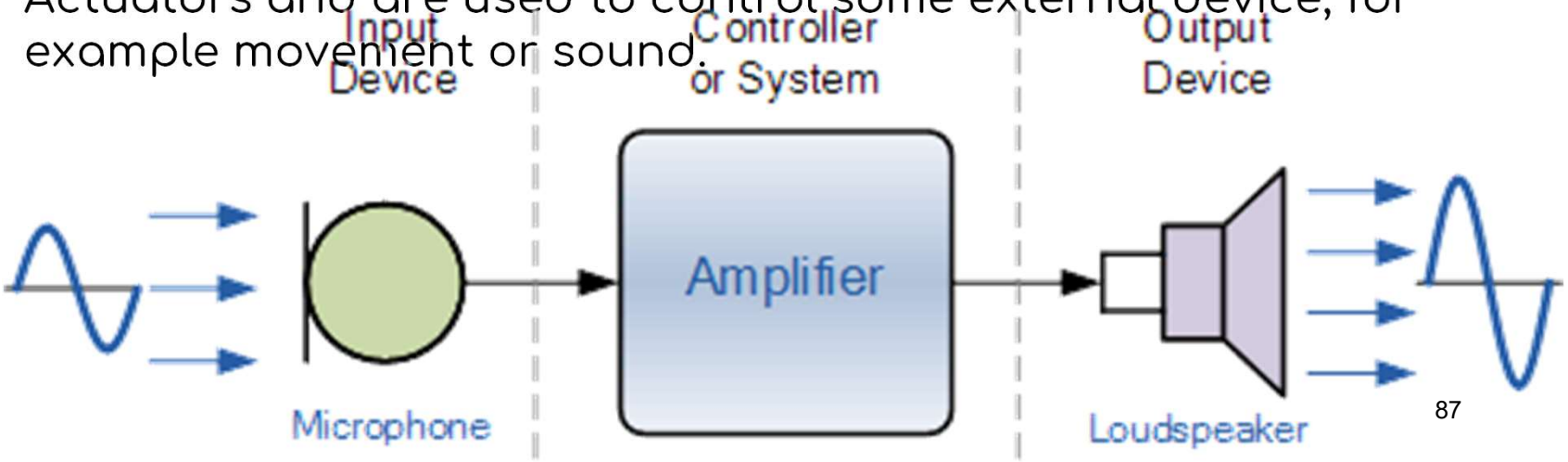
Hint:

```
int LED = 3;
int brightness = 0;
int fadeAmount = 5;
void setup()
{ pinMode(3, OUTPUT); }
void loop()
{
 analogWrite(LED, brightness);
 brightness = brightness +
fadeAmount;
 if (brightness <= 0 || brightness >= 255)
 {
 fadeAmount = -fadeAmount;
 }
 delay(30); }
```

Topic 3
Analog Sensors and Transducers

Sensors and Transducers

- In order for an electronic circuit or system to perform any useful task or function it needs to be able to communicate with the "real world"
- The word "Transducer" is the collective term used for both Sensors which can be used to sense a wide range of different energy forms such as movement, electrical signals, radiant energy, thermal and Actuators which can be used to switch voltages or currents.
- There are many different types of sensors and transducers, both analogue and digital and input and output available to choose from.
- Devices which perform an "Input" function are commonly called Sensors
- Devices which perform an "Output" function are generally called Actuators and are used to control some external device, for example movement or sound.

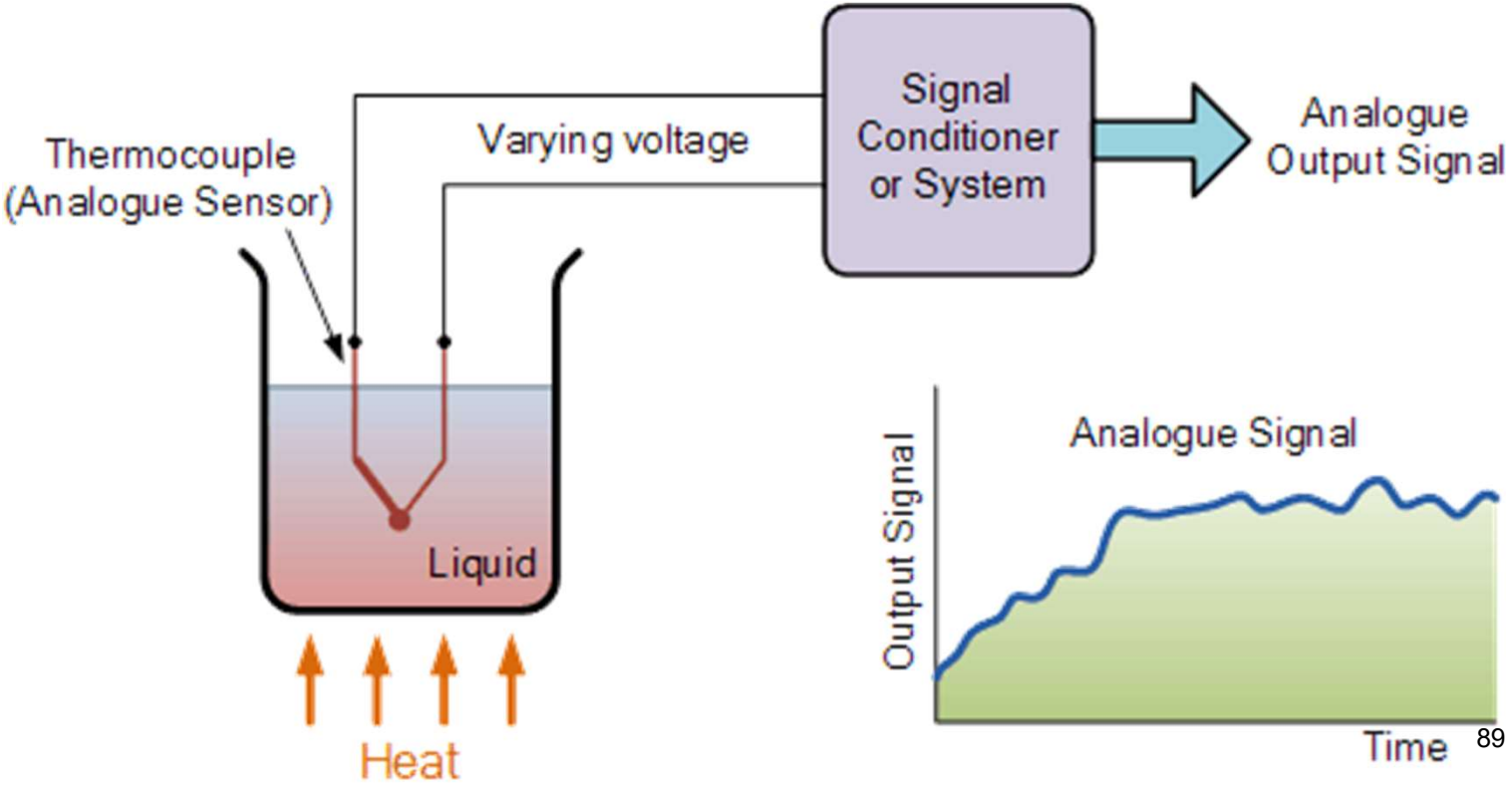


Common Sensors and Transducers

Quantity being Measured	Input Device (Sensor)	Output Device (Actuator)
Light Level	Light Dependant Resistor (LDR) Photodiode Photo-transistor Solar Cell	Lights & Lamps LED's & Displays Fibre Optics
Temperature	Thermocouple Thermistor Thermostat Resistive Temperature Detectors	Heater Fan
Force/Pressure	Strain Gauge Pressure Switch Load Cells	Lifts & Jacks Electromagnet Vibration
Position	Potentiometer Encoders Reflective/Slotted Opto-switch LVDT	Motor Solenoid Panel Meters
Speed	Tacho-generator Reflective/Slotted Opto-coupler Doppler Effect Sensors	AC and DC Motors Stepper Motor Brake
Sound	Carbon Microphone Piezo-electric Crystal	Bell Buzzer Loudspeaker

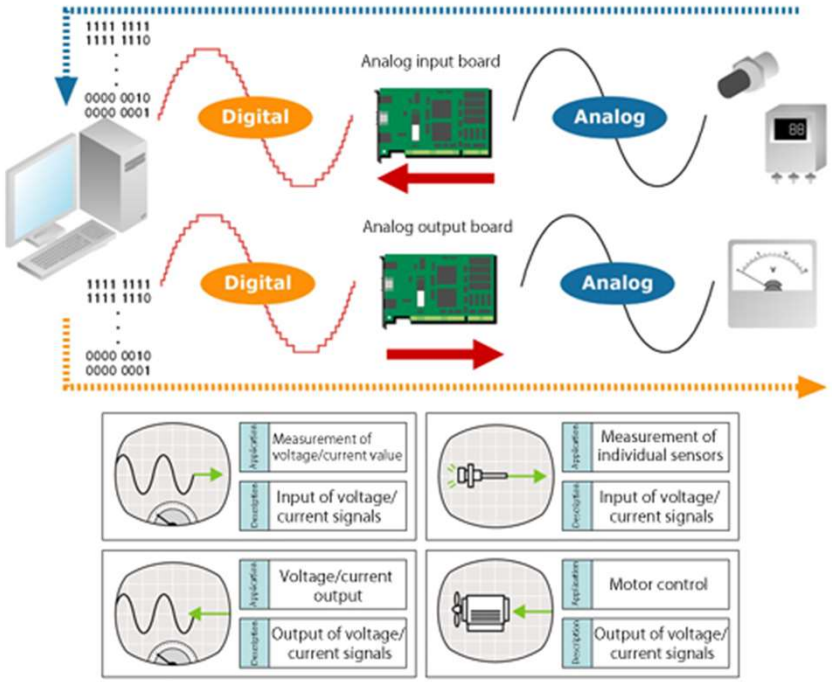
Analogue Sensors

- Analogue Sensors produce a continuous output signal or voltage which is generally proportional to the quantity being measured. Physical quantities such as Temperature, Speed, Pressure, Displacement, Strain etc are all analogue quantities as they tend to be continuous in nature.
- For example, the temperature of a liquid can be measured using a thermometer or thermocouple which continuously responds to temperature changes as the liquid is heated up or cooled down.



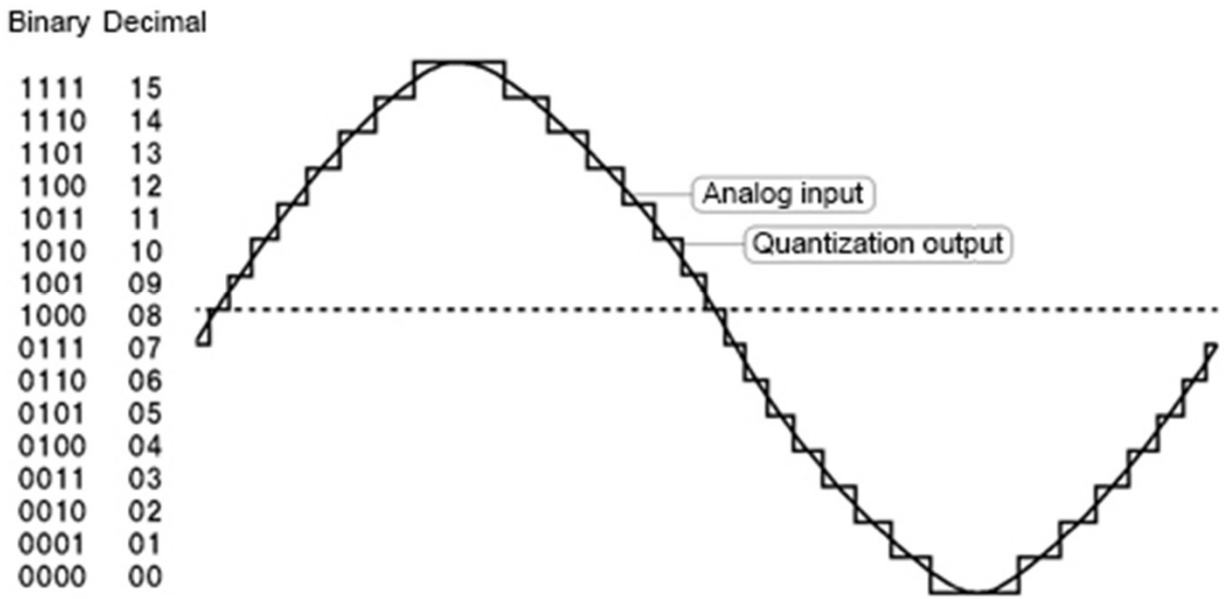
What is Analog I/O?

- The signals from sensors that measure surrounding natural factors such as temperature, pressure, and flow rate are often analog signals, and most control actuators move according to analog signals.
- On the other hand, only digital signals can be handled by computers.
- For this reason, in order to input a signal from a sensor using a computer, or to output a signal to an actuator, it's necessary to have a device that can bridge the analog signal and the digital signal handled by the computer. That bridge is called an analog I/O interface.



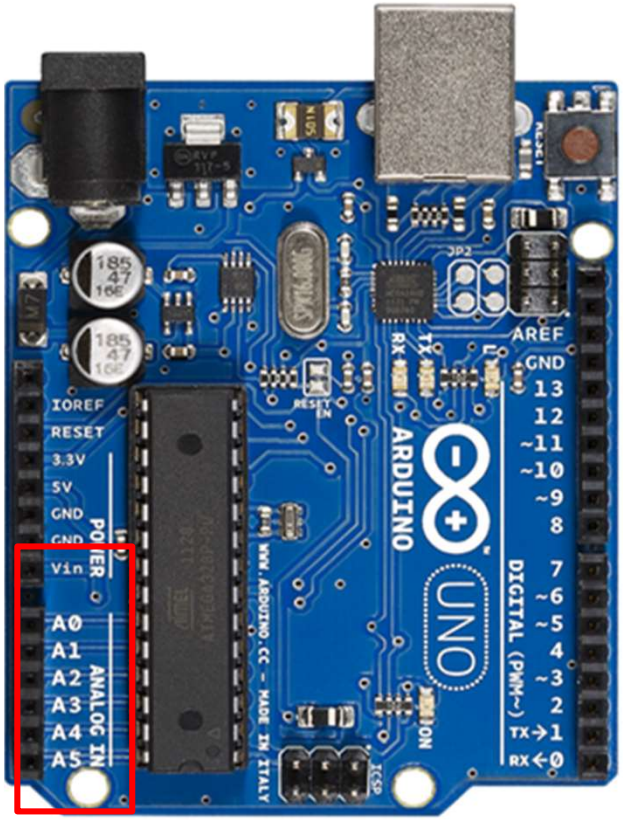
Analog to Digital Conversion

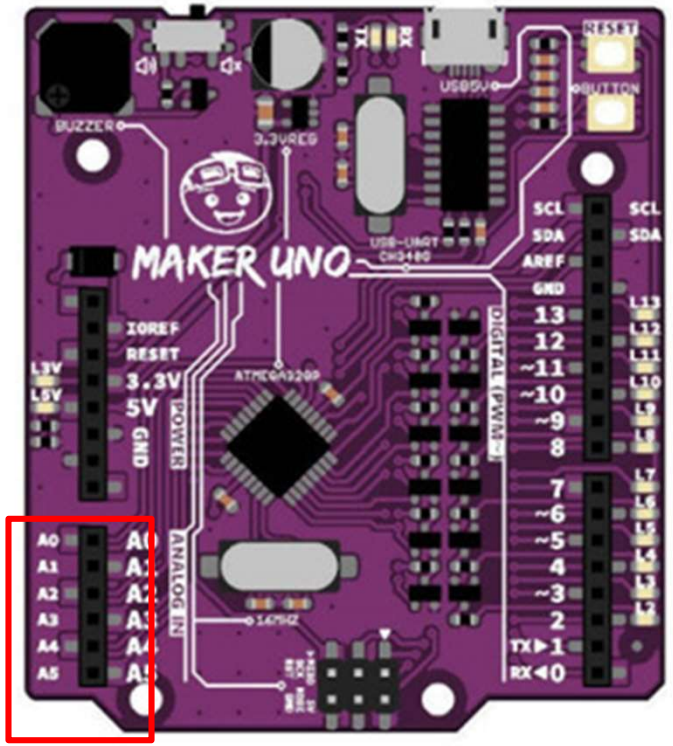
- When quantizing analog values represented by a solid line, you get a stepped line. This makes it possible to express any analog signal using a finite value.
- This technology is active in things many people are familiar with, for example, in cellular phones. Cell phones make calls by converting voices (analog) to digital sounds



Analog Pins

- A few PINs on the Arduino Analog PINs allow read analog signal
- A0, A1, A2, A3, A4, A5





AnalogRead

- Writes an analog value (PWM wave) to a pin. Can be used to light a LED at varying brightnesses or drive a motor at various speeds.
- After a call to analogWrite(), the pin will generate a steady rectangular wave of the specified duty cycle until the next call to analogWrite() (or a call to digitalRead() or digitalWrite()) on the same pin.

- Syntax
- analogRead(PIN)

- For example
- analogRead(A0)

Map

- Re-maps a number from one range to another. That is, a value of from Low would get mapped to Low, a value of from High to to High, values in-between to values in-between, etc.

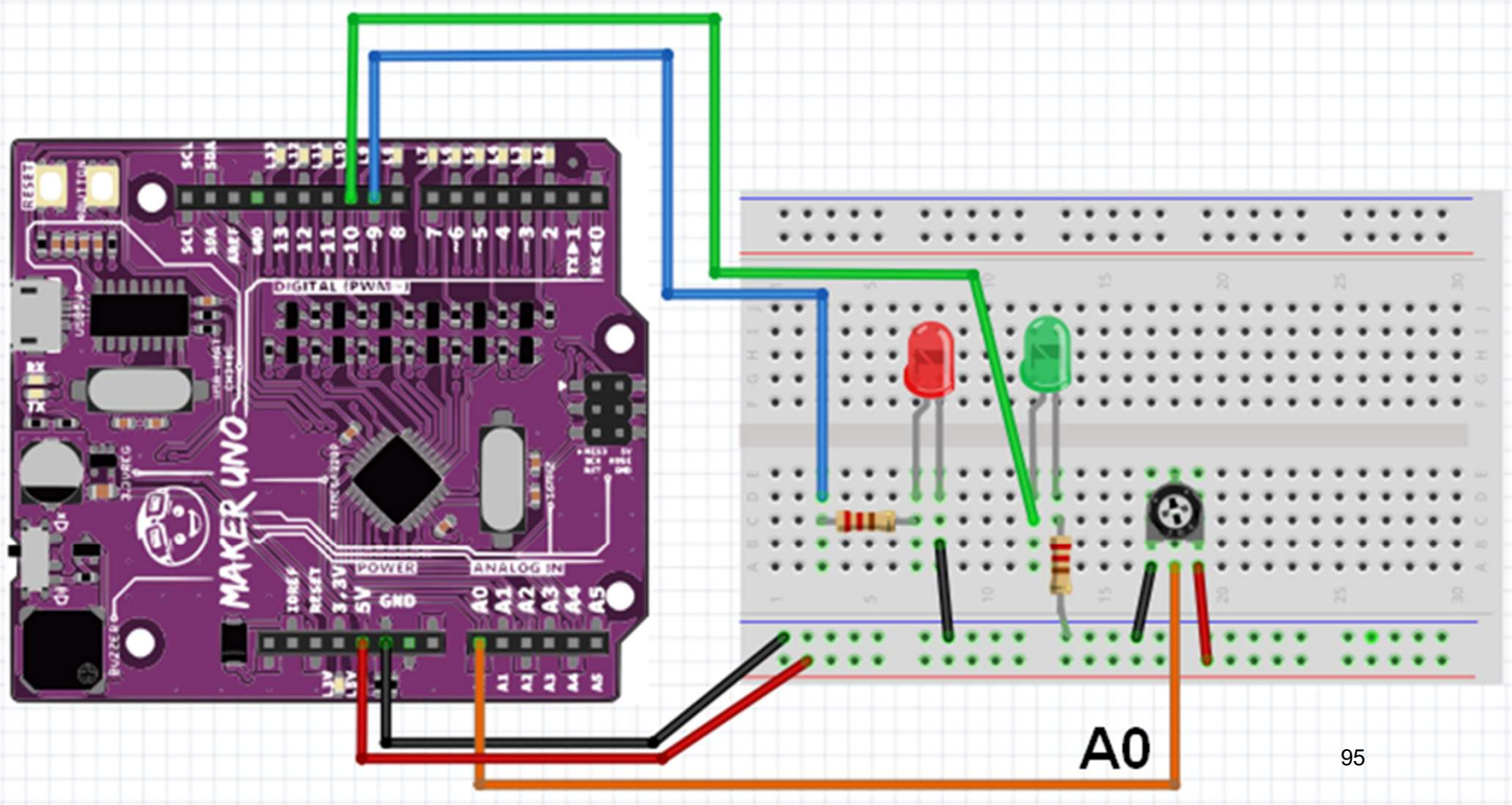
- Syntax:
- map(value, fromLow, fromHigh, toLow, toHigh)

- For Example:

```
int val = analogRead(0);
val = map(val, 0, 1023, 0, 255);
analogWrite(9, val);
```

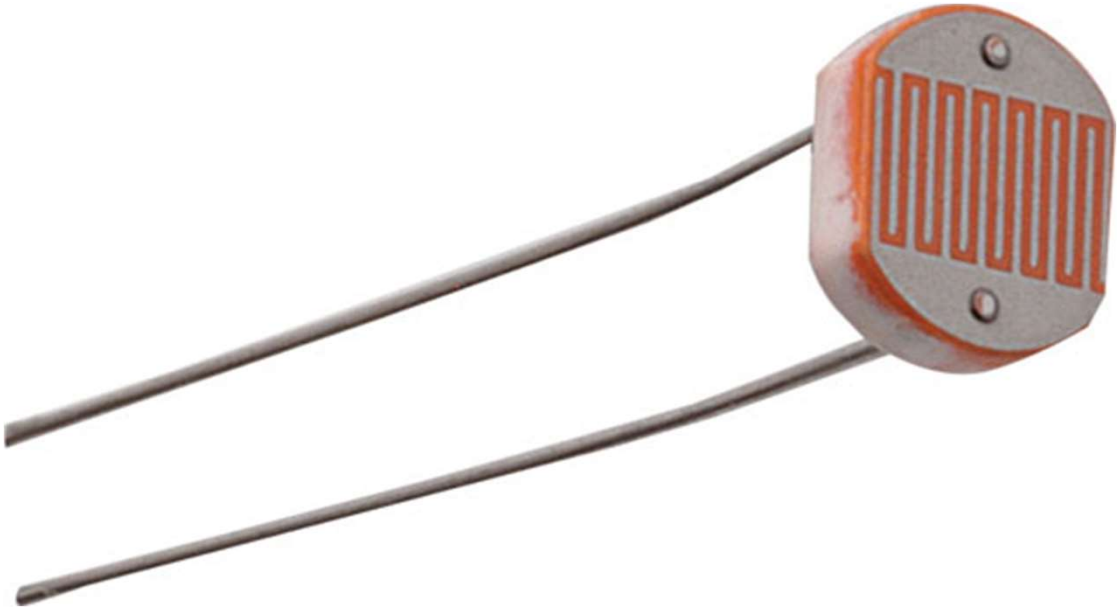
Activity: Analog I/O

- Code a sketch to vary the brightness of the LED using potentiometer.
- Print out the analog brightness value on the serial monitor



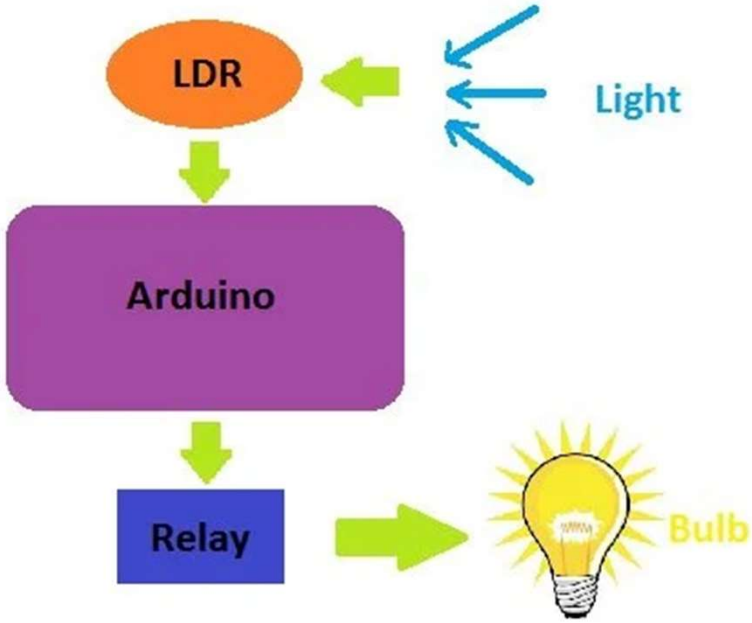
Detecting Light Intensity

- To detect the intensity of light or darkness, we use a sensor called an LDR (light dependent resistor).
- The LDR is a special type of resistor that allows higher voltages to pass through it (low resistance) whenever there is a high intensity of light, and passes a low voltage (high resistance) whenever it is dark



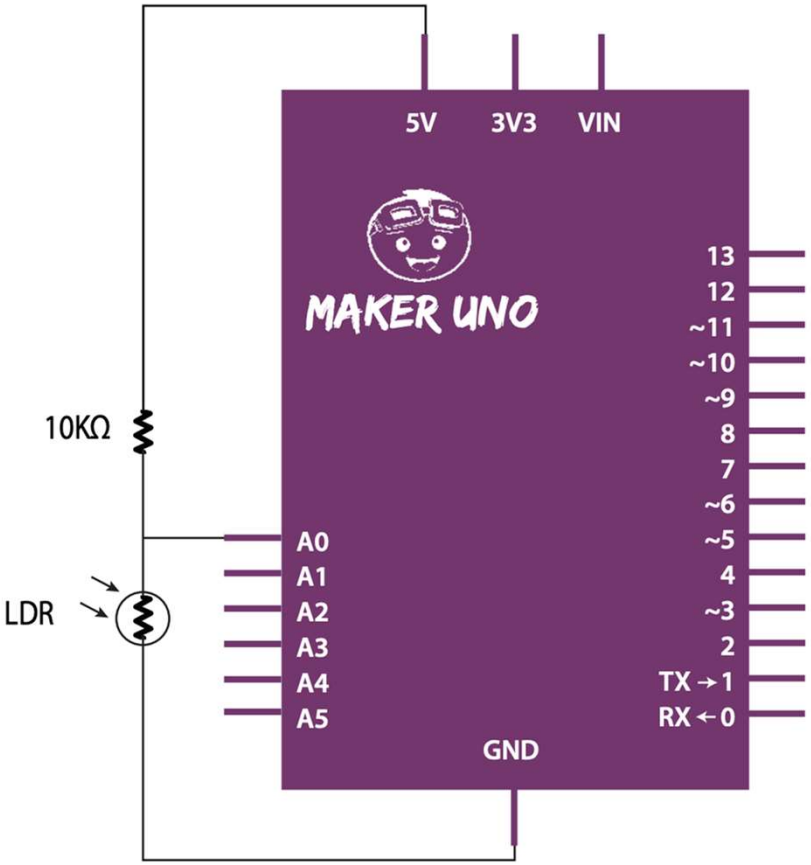
How Does LDR Work?

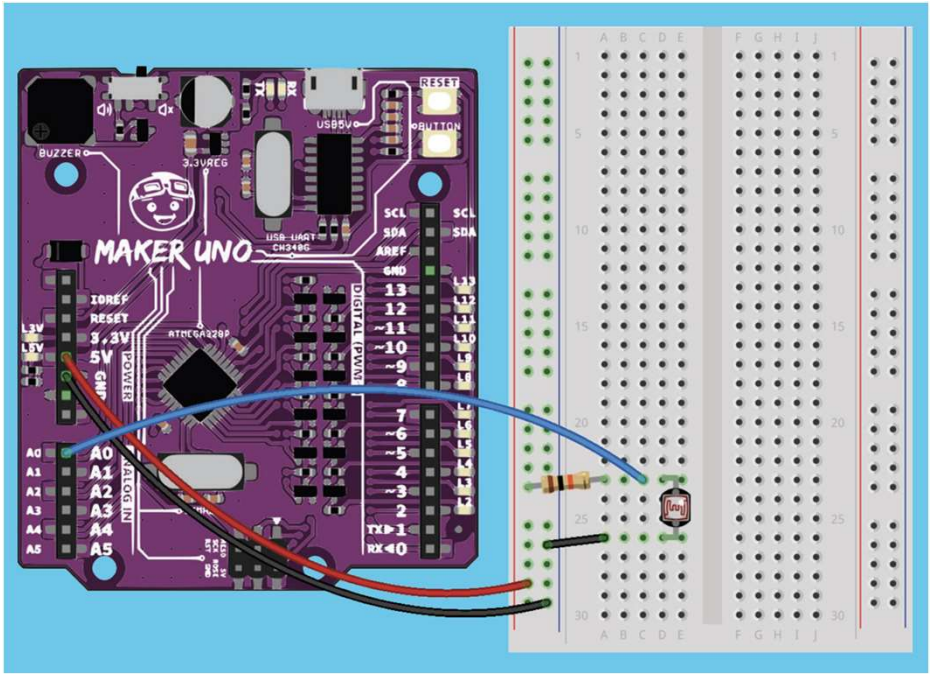
- The LDR gives out an analog voltage when connected to VCC (5V), which varies in magnitude in direct proportion to the input light intensity on it.
- That is, the greater the intensity of light, the greater the corresponding voltage from the LDR will be.
- Since the LDR gives out an analog voltage, it is connected to the analog input pin on the Arduino.
- The Arduino, with its built-in ADC (analog-to-digital converter), then converts the analog voltage (from 0-5V) into a digital value in the range of (0-1023).



Activity: LDR Analog Input

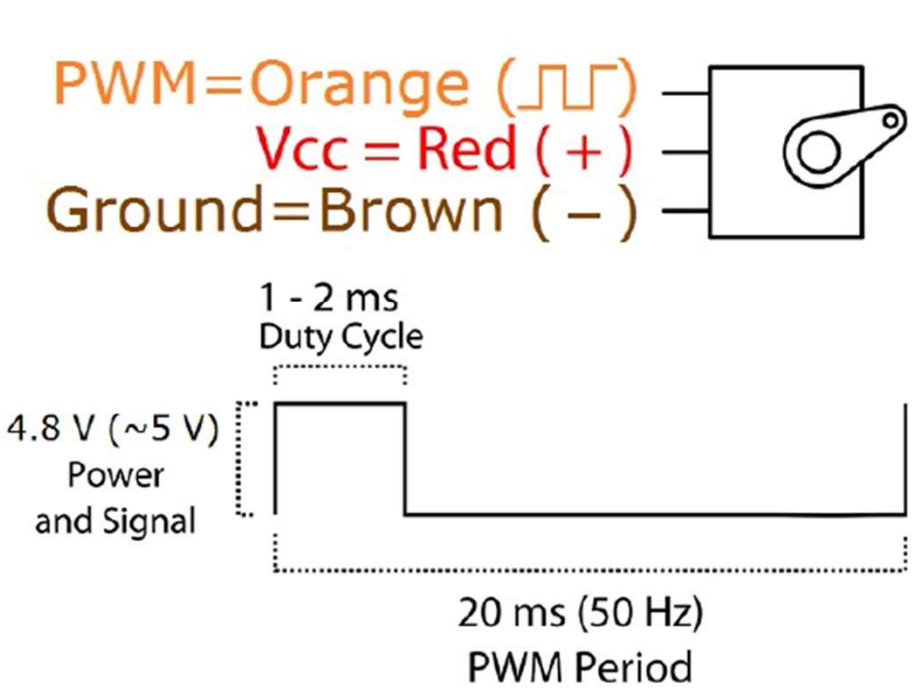
- Connect the circuit as shown below
- Write a sketch to read the analog output of a LDR
- When it is dark, the LED on pin 13 will light up

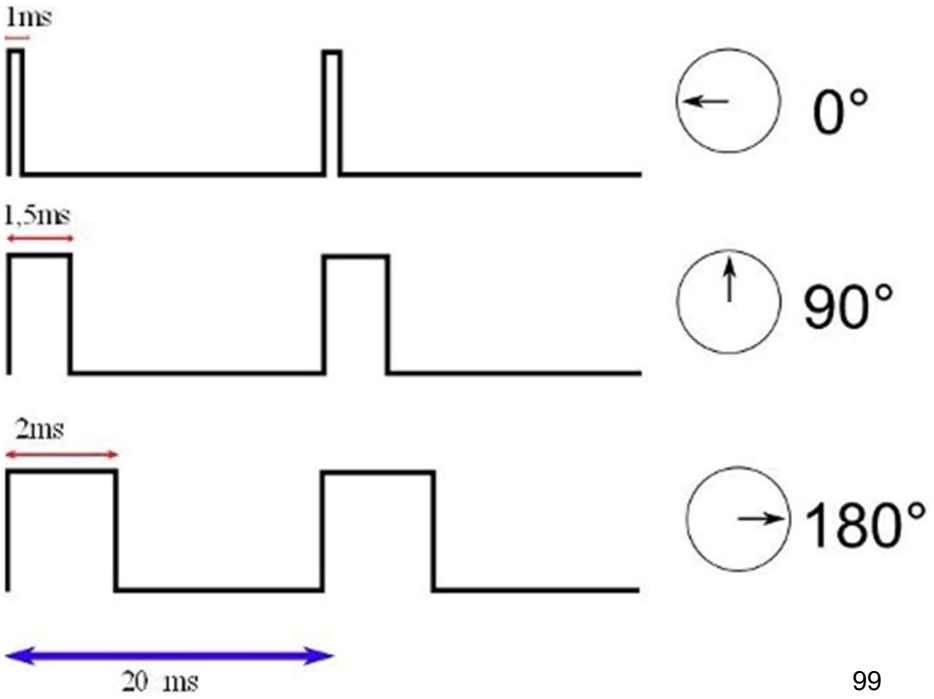




Servo Motor

- Geared motor
- Rotate 0° to 180°
- 3 pins
- PWM signal



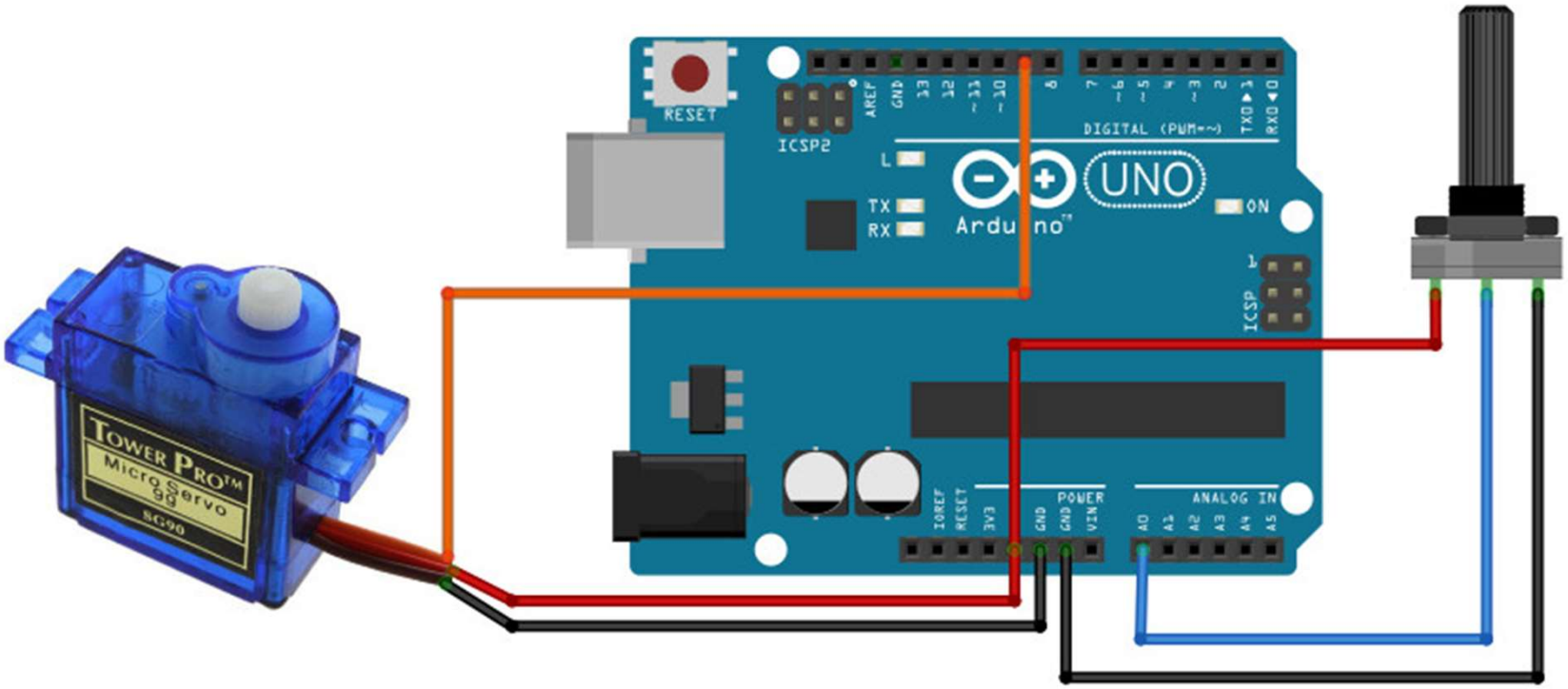


Servo Motor Sketch Example

```
#include <Servo.h>
Servo myservo;  // create servo object to control a servo

int potpin = 0;  // analog pin used to connect the potentiometer
int val;    // variable to read the value from the analog pin

void setup() {
    myservo.attach(9);  // attaches the servo on pin 9 to the servo object
   }

void loop() {
  val = analogRead(potpin);
  val = map(val, 0, 1023, 0, 180);
  myservo.write(val);           // sets the servo position according to the scaled value
   delay(15);                   // waits for the servo to get there
    }
```

Activity: Servo Motor

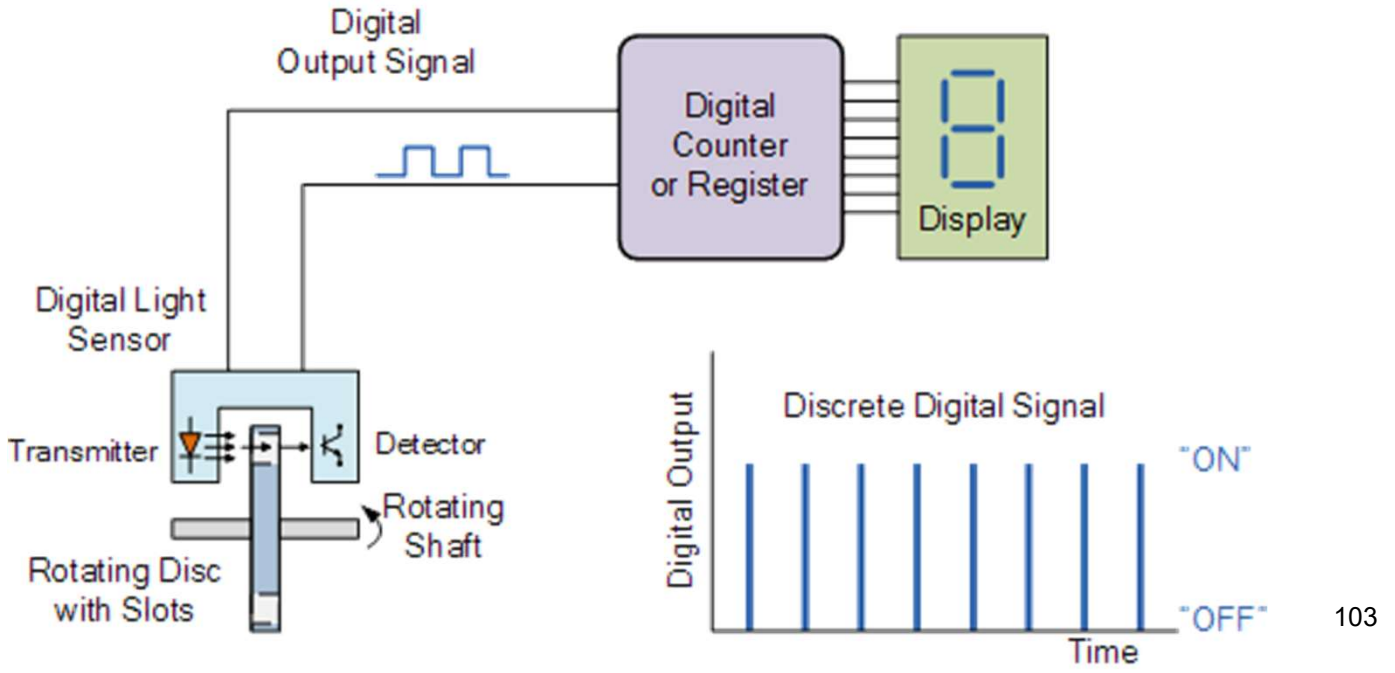
- Connect Servo Motor to the Arduino
- Use the potentiometer to vary the angle from 0 to 180 degrees.



Topic 4
Digital Sensors and Transductors

Digital Sensors

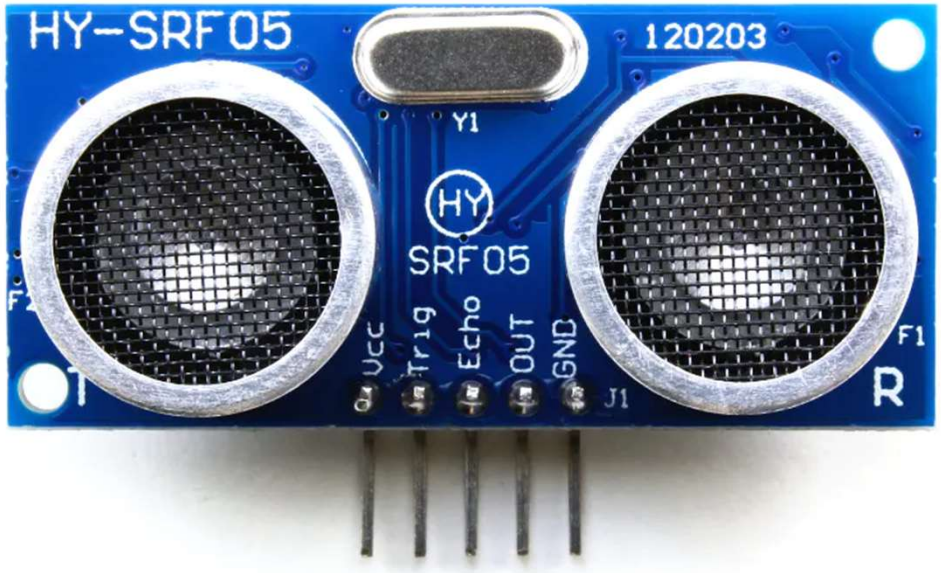
- As its name implies, Digital Sensors produce a discrete digital output signals or voltages that are a digital representation of the quantity being measured. Digital sensors produce a Binary output signal in the form of a logic "1" or a logic "0", ("ON" or "OFF").
- For example, the speed of the rotating shaft is measured by using a digital LED/Opto-detector sensor. The disc which is fixed to a rotating shaft (for example, from a motor or robot wheels), has a number of transparent slots within its design. As the disc rotates with the speed of the shaft, each slot passes by the sensor in turn producing an output pulse representing a logic "1" or logic "0" level.



Ultrasonic Sensor HY-SRF05

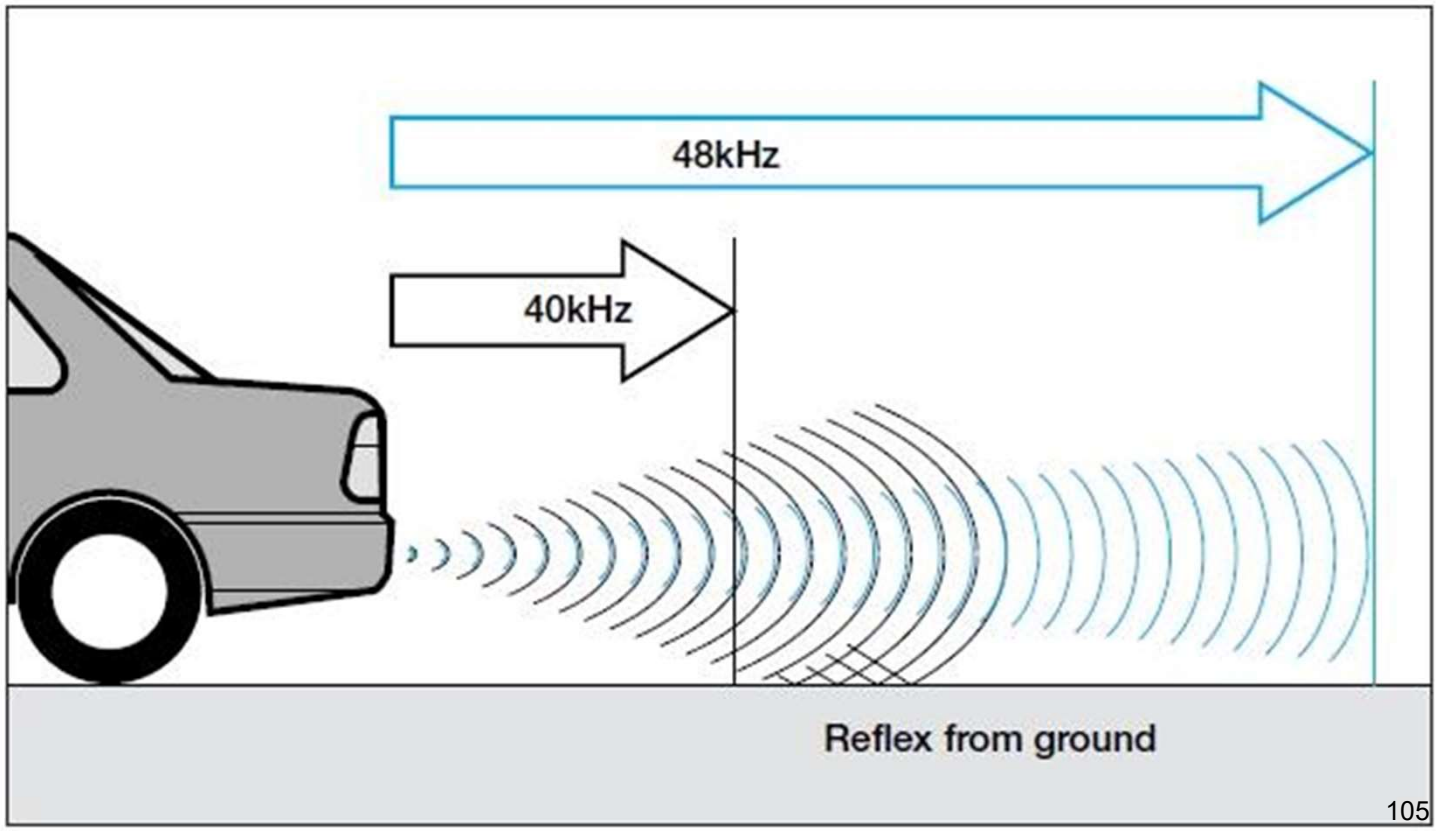
- The ultrasonic sensor measures the distance of the nearest object, sending the result to the serial port. It can work from 2 cm to 3 m. It measures the time spent by the signal to reach the object and return to the sensor.
- Connections:

Vcc -> 5 V
Trig -> pin 13 (digital pin)
Echo -> pin 12 (digital pir
Out ->
GND -> GND



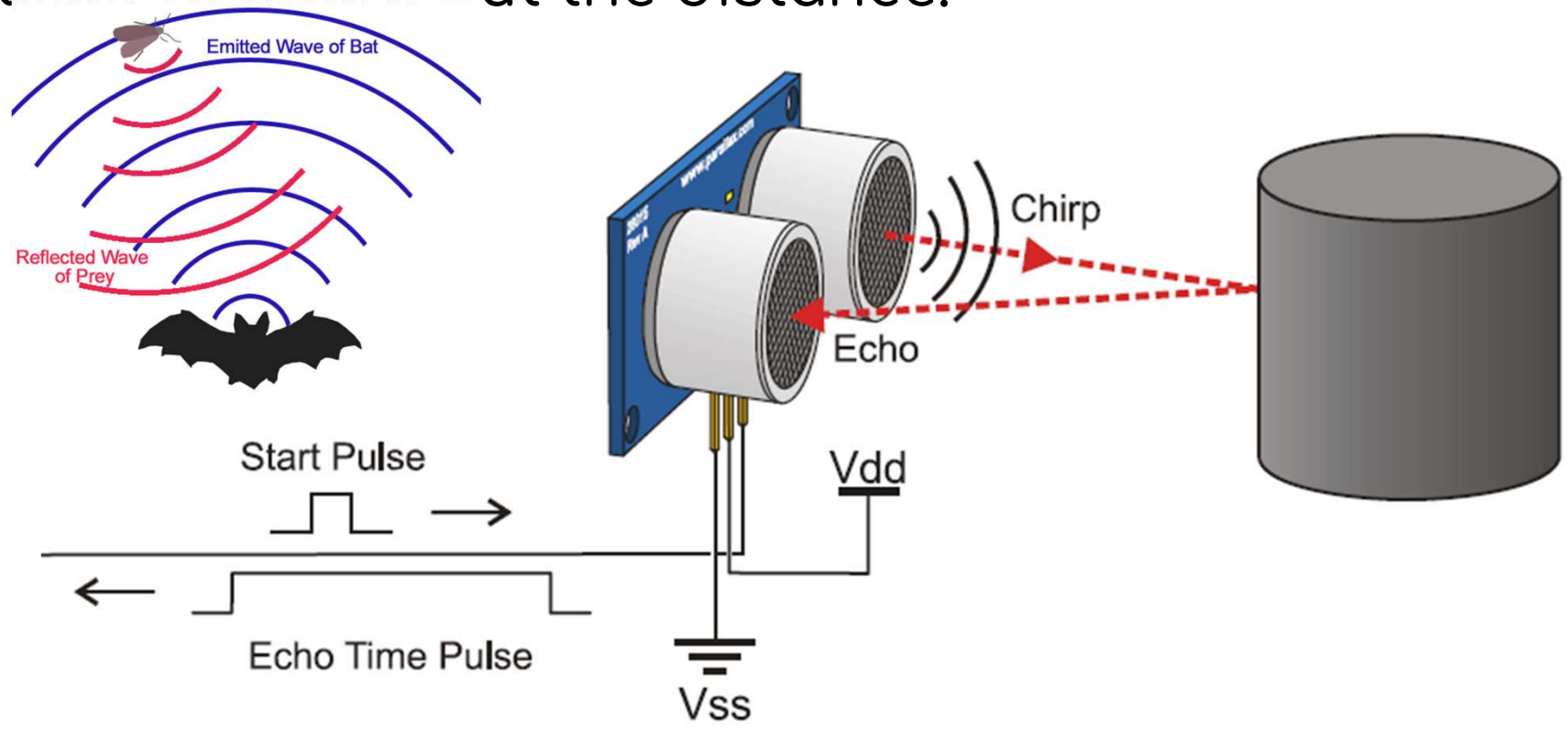
Application of Ultrasonic Sensor

- Used at rear car bumper
- Inaudible to human
- Insensitive to surrounding lighting
- Work under Sunlight and darkness



How Ultrasonic Sensor Work?

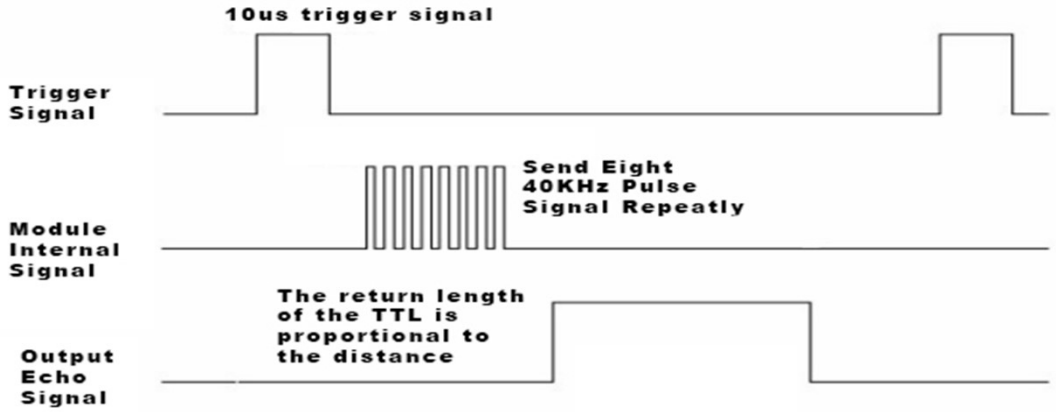
- The ultrasonic sensor works like a bat echolocation.
- A chirp is emitted from the transmitter
- It bounces off of an object.
- The echo returns to the microphone.
- The time it takes to travel to the object and back is used to figure out the distance.



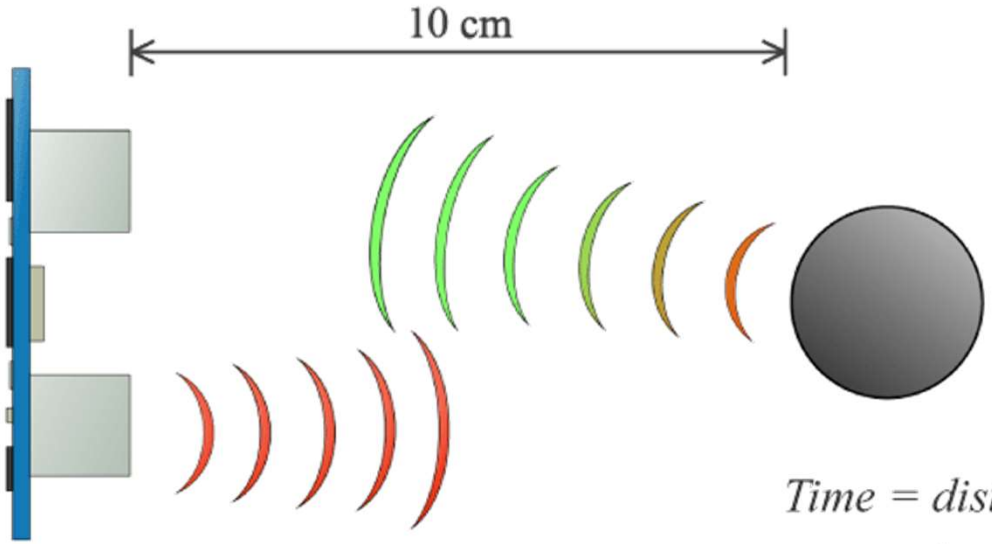
Initiate HY-SRF05

- Supply a short 10uS pulse to the trigger input to start the ranging.
- Send out an 8 cycle burst of ultrasound at 40khz
- Raise its echo line high
- Listens for an echo, once it detects one it lowers the echo line again
- Echo line is therefore a pulse whose width is proportional to the distance to the object.

```
digitalWrite(TRIG_PIN, LOW);
delayMicroseconds(2);
digitalWrite(trigPin, HIGH);
delayMicroseconds(10);
digitalWrite(trigPin, LOW);
```



Distance Calculation



speed of sound:

$v = 340\ m/s$

$v = 0{,}034\ cm/\mu s$

Time = distance / speed:

$t = s / v = 10 / 0{,}034 = 294\ \mu s$

Distance:

$s = t \cdot 0{,}034 / 2$

```
duration = pulseIn(echoPin, HIGH);
distance= duration*0.034/2;
```

PulseIn

- Reads a pulse (either HIGH or LOW) on a pin
- For example, if value is HIGH, pulseIn() waits for the pin to go from LOW to HIGH, starts timing, then waits for the pin to go LOW and stops timing.
- Returns the length of the pulse in microseconds or gives up and returns 0 if no complete pulse was received within the timeout.

Syntax

pulseIn(pin, value)

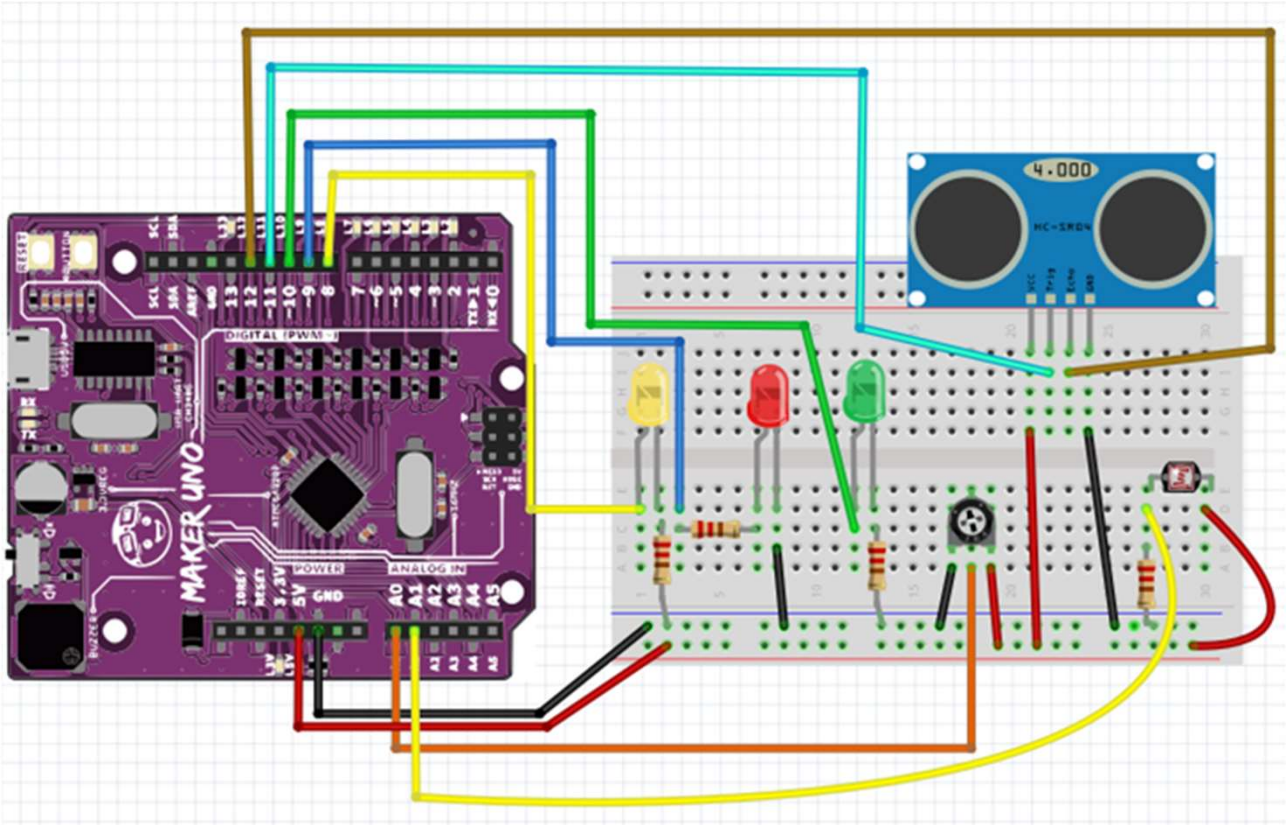
pulseIn(pin, value, timeout)

Sketch to Compute Distance

```
// Sets the trigPin on HIGH state for 10 micro seconds
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  // Reads the echoPin, returns the sound wave travel
time in microseconds
  duration = pulseIn(echoPin, HIGH);

  // Calculating the distance
  distance= duration*0.034/2;
```

Activity: Motion/Obstacle Detection

- Wired a obstacle detection system.
- When there is an object blocking in front, the RED LED light will turn on.
- When no object is blocking, the Green LED will switch on



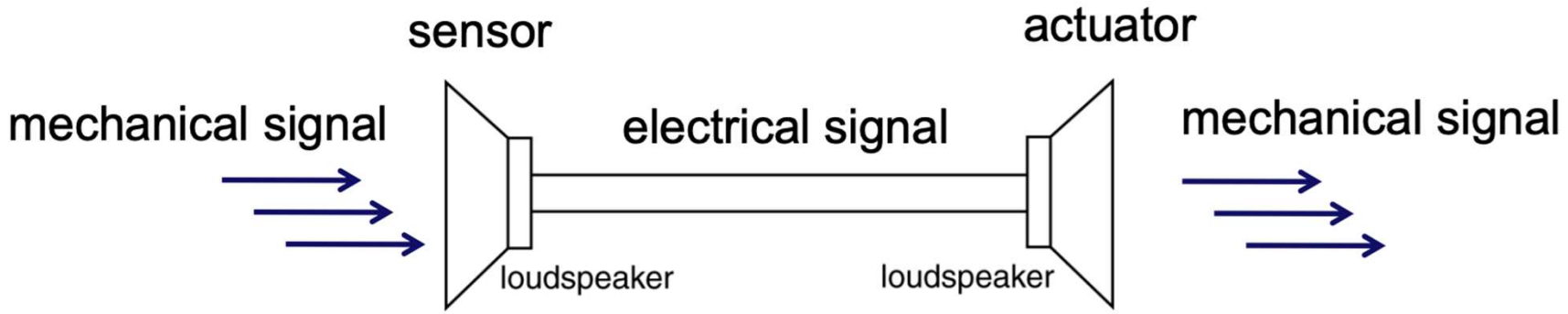
Topic 5
Actuators

Actuator

- An actuator is a component of a machine that is responsible for moving and controlling a mechanism or system, for example by opening a valve. In simple terms, it is a "mover".
- An actuator requires a control signal and a source of energy. The control signal is relatively low energy and may be electric voltage or current, pneumatic or hydraulic pressure, or even human power. Its main energy source may be an electric current, hydraulic fluid pressure, or pneumatic pressure. When it receives a control signal, an actuator responds by converting the source's energy into mechanical motion.

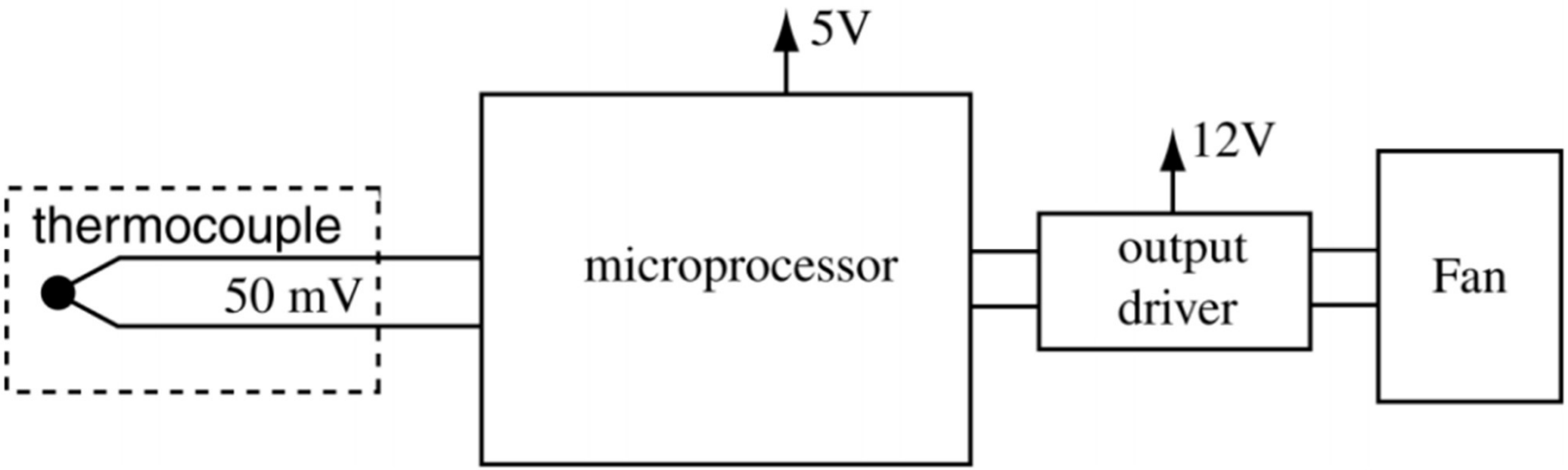
Example - Loudspeaker

- A transducer converts a stimulus from a signal domain to another signal domain
- A sensor receives a stimulus and responds with an electrical signal
- An actuator converts an electrical signal to another signal domain



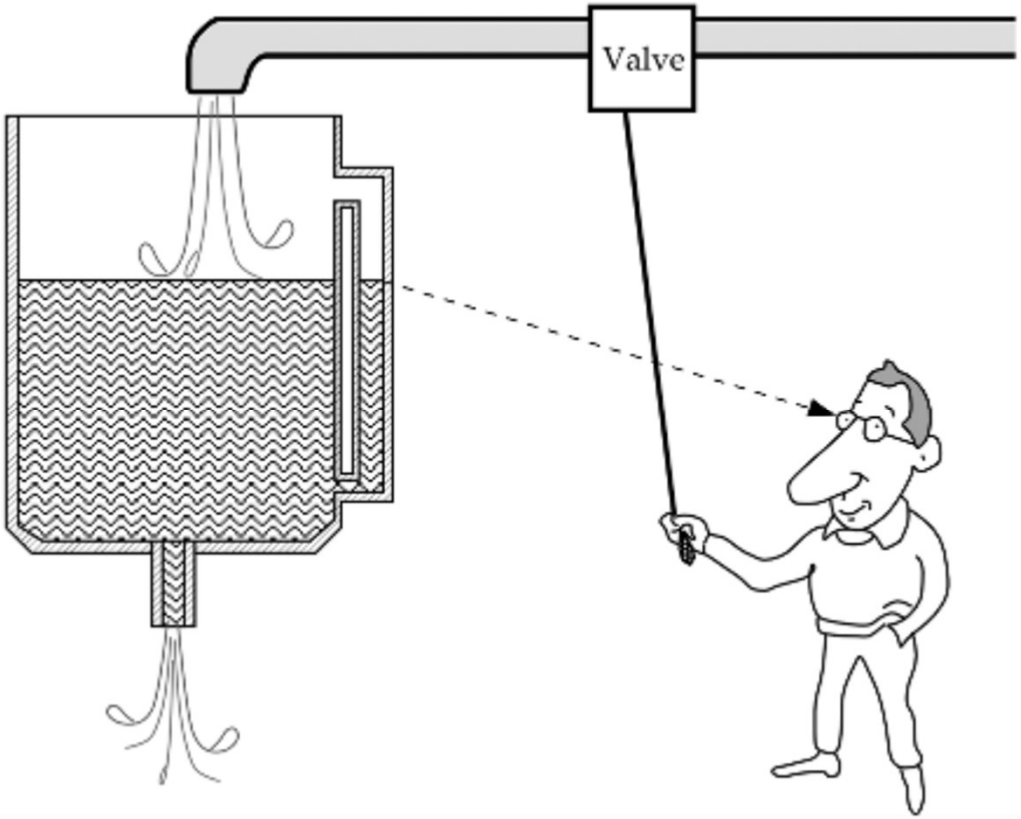
Example - Temperature Control

- Sense the temperature of a CPU
- Control the speed of the fan to keep the temperature constant



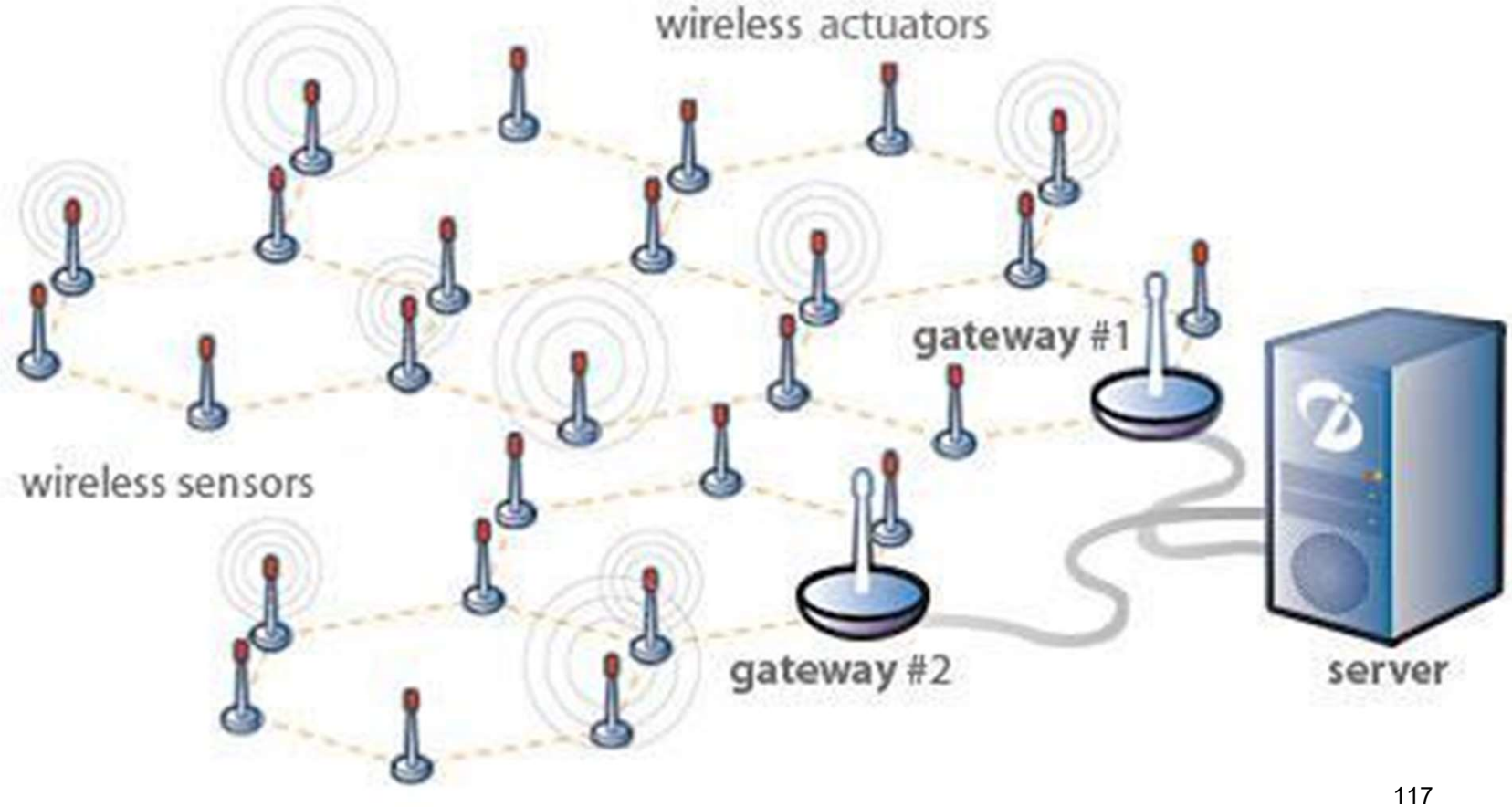
Example - Level Control

- Sense the water level
- Control the valve of the thank to keep the water level constant



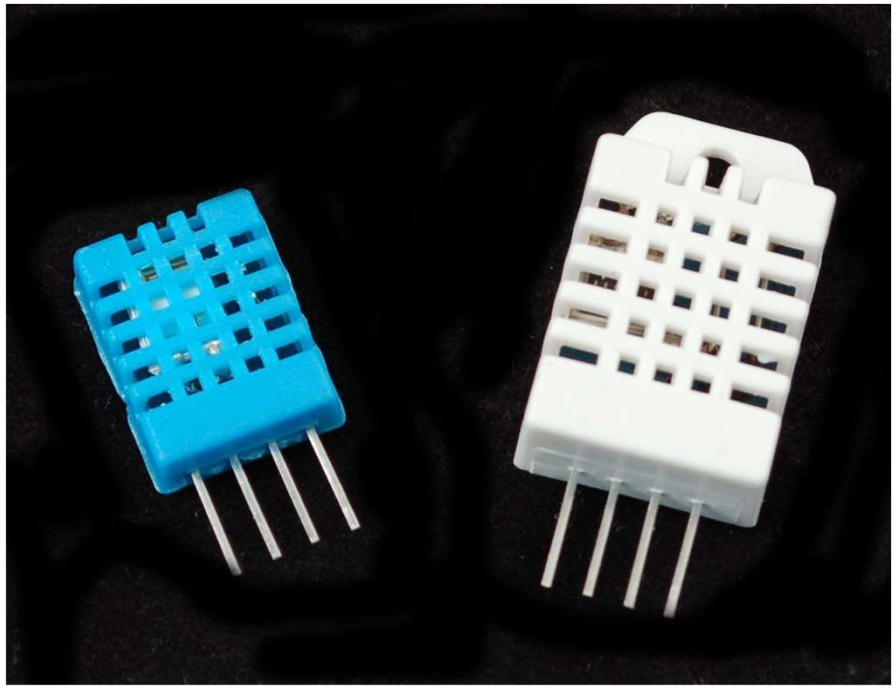
Wireless Actuator Network

- With the advent of IoT, Wireless Sensor & Actuator Network (WSAN) is becoming popular. e



DHT Sensors

- The DHT sensors are made of two parts, a capacitive humidity sensor and a thermistor.
- There is also a very basic chip inside that does some analog to digital conversion and spits out a digital signal with the temperature and humidity.
- There are two popular DHT sensor: DHT11 and DHT22



DHT11

- Ultra low cost
- 3 to 5V power and I/O
- 2.5mA max current use during conversion (while requesting data)
- Good for 20-80% humidity readings with 5% accuracy
- Good for 0-50°C temperature readings ±2°C accuracy
- No more than 1 Hz sampling rate (once every second)
- Body size 15.5mm x 12mm x 5.5mm
- 4 pins with 0.1" spacing

DHT22

- Low cost
- 3 to 5V power and I/O
- 2.5mA max current use during conversion (while requesting data)
- Good for 0-100% humidity readings with 2-5% accuracy
- Good for -40 to 80°C temperature readings ±0.5°C accuracy
- No more than 0.5 Hz sampling rate (once every 2 seconds)
- Body size 15.1mm x 25mm x 7.7mm
- 4 pins with 0.1" spacing

What is Relative Humidity?

- The DHT11 measures relative humidity. The relative humidity is the amount of water vapor in air vs. the saturation point of water vapor in the air. At the saturation point, water vapor starts to condense and accumulate on surfaces forming dew.
- The saturation point changes with air temperature. Cold air can hold less water vapor before it becomes saturated, and hot air can hold more water vapor before it becomes saturated.
- The formula to calculate relative humidity is:

$$RH = \left(\frac{\rho_w}{\rho_s}\right) x\ 100\%$$

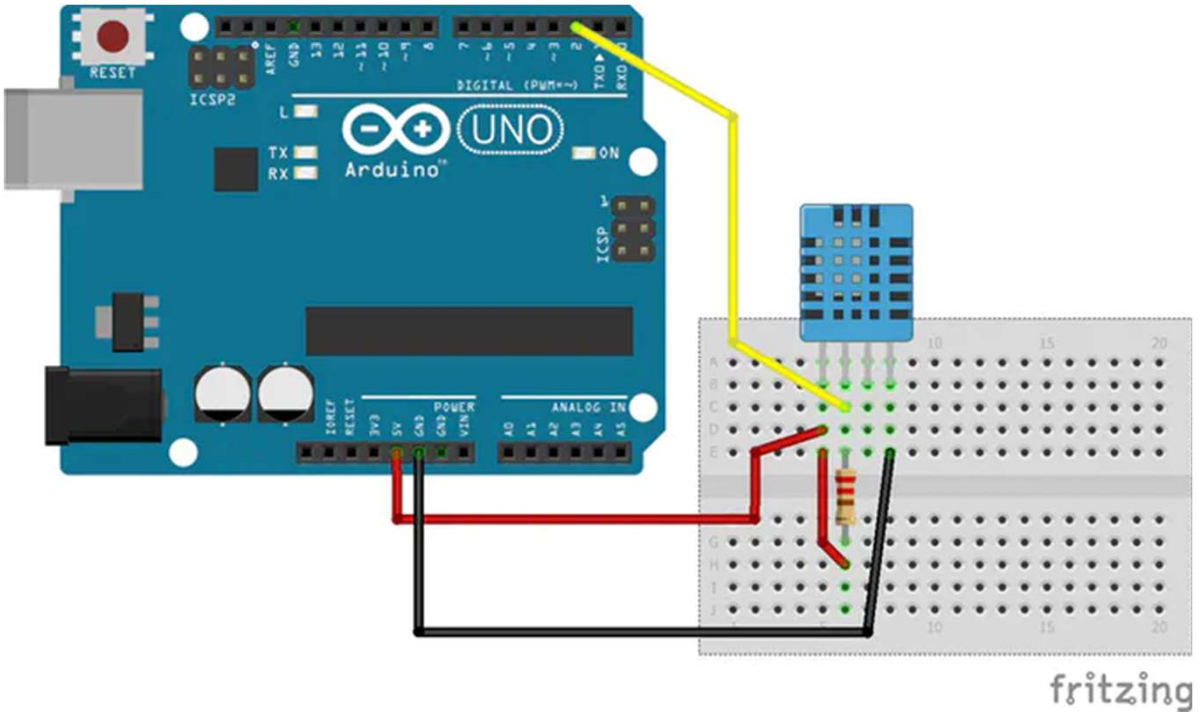
RH : *Relative Humidity*
ρ_w : *Density of water vapor*
ρ_s : *Density of water vapor at saturation*

How the DHT11 Measures Humidity and Temperature

- The DHT11 detects water vapor by measuring the electrical resistance between two electrodes.
- The humidity sensing component is a moisture holding substrate with electrodes applied to the surface. When water vapor is absorbed by the substrate, ions are released by the substrate which increases the conductivity between the electrodes.
- The change in resistance between the two electrodes is proportional to the relative humidity. Higher relative humidity decreases the resistance between the electrodes, while lower relative humidity increases the resistance between the electrodes.
- The DHT11 measures temperature with a surface mounted NTC temperature sensor (thermistor) built into the unit.

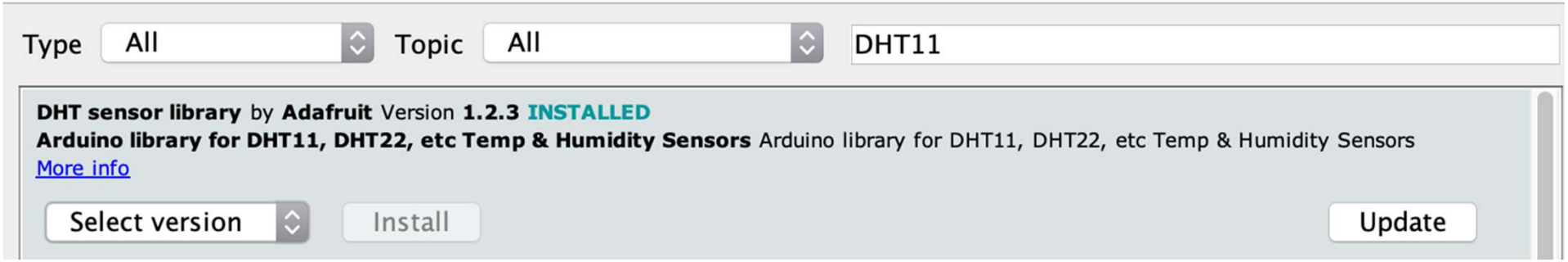
Set Up DHT11 on an Arduino

- VCC - red wire Connect to 3.3 - 5V power. Sometime 3.3V power isn't enough in which case try 5V power.
  - Data out - white or yellow wire
  - Not connected
  - Ground - black wire
- Simply ignore pin 3, it's not used. You will want to place a 10 K ohm resistor between VCC and the data pin, to act as a medium-strength pull up on the data line. The Arduino has built-in pull-ups you can turn on but they're very weak, about 20-50K



Install DHT Library

- Goto Sketch -> Include Library -> Manage Libraries
- Install DHT sensor library by Adafruit. Recommended version is 1.2.3

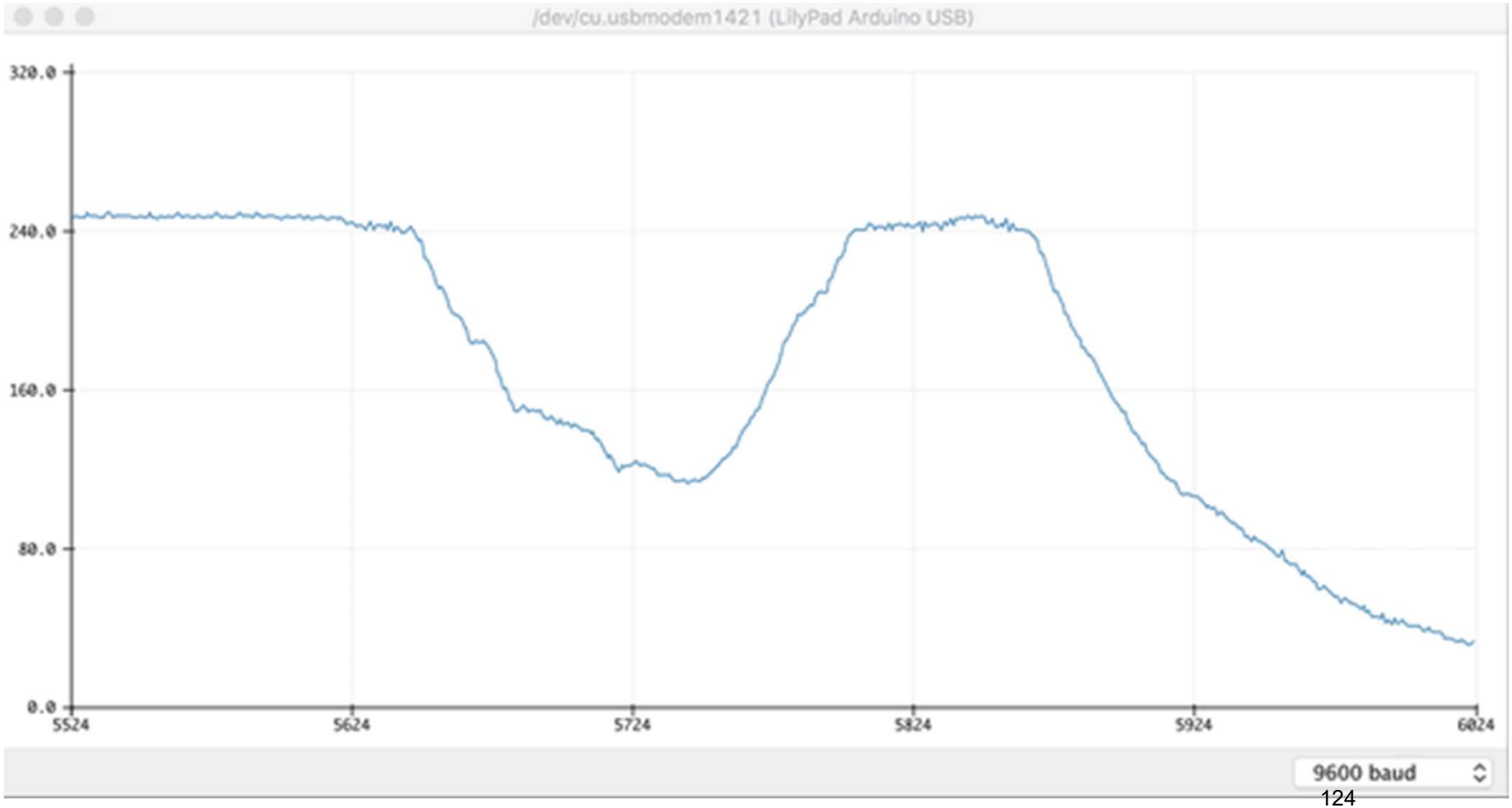


Use DHT Library

```
#include <DHT.h>                                    //Import
DHT Library

#define DHTPIN 12                                   //
Should be GPIO #
#define DHTTYPE DHT11                           // DHT11 or
DHT22
DHT dht(DHTPIN, DHTTYPE);                   //Create DHT object


dht.readTemperature()                           //Read
Temperature
dht.readHumidity()                                  //Read
Humidity
```

Activity: DHT

- Code a sketch to monitor the temperature and humidity data on the serial plotter.

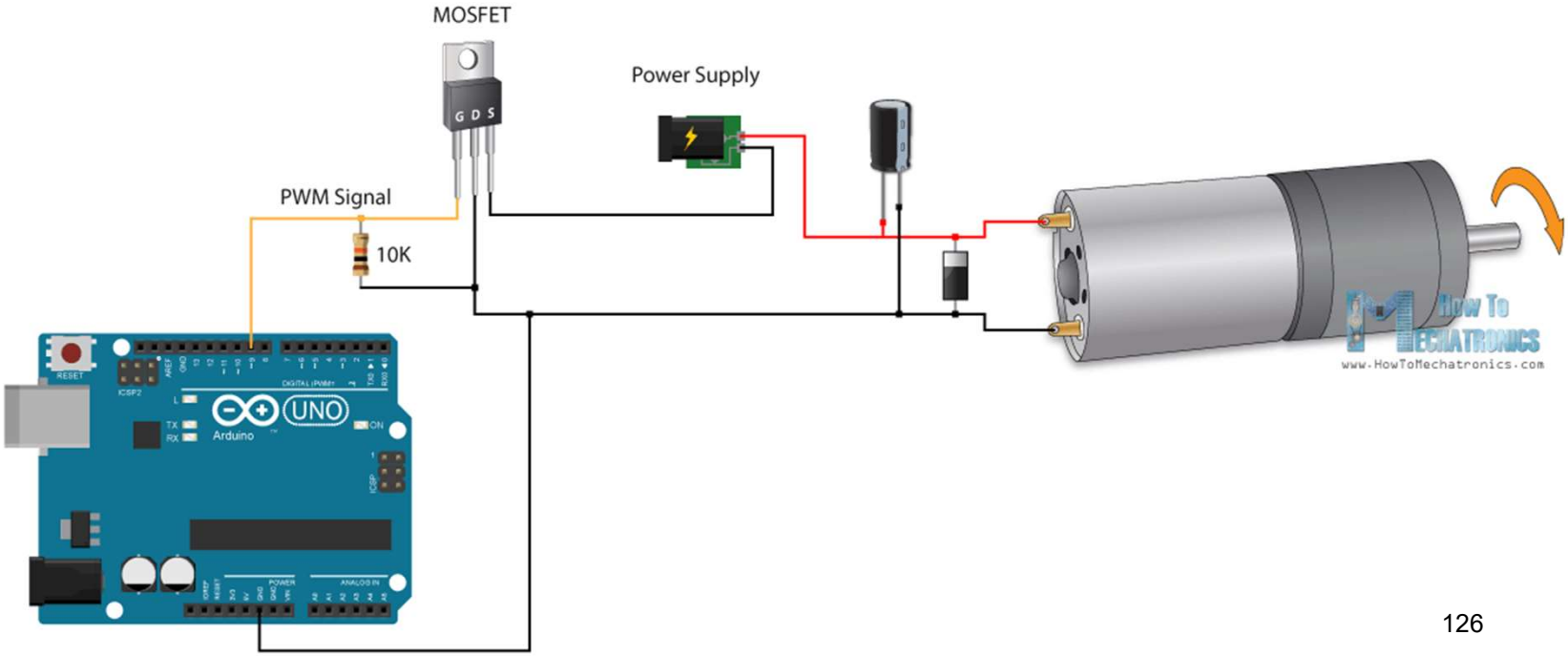


DC Motor

- Used in many portable home appliances, automobiles and industrial equipment
- Works on the principle that when a current carrying conductor is placed in a magnetic field, it experiences a torque and has a tendency to move.

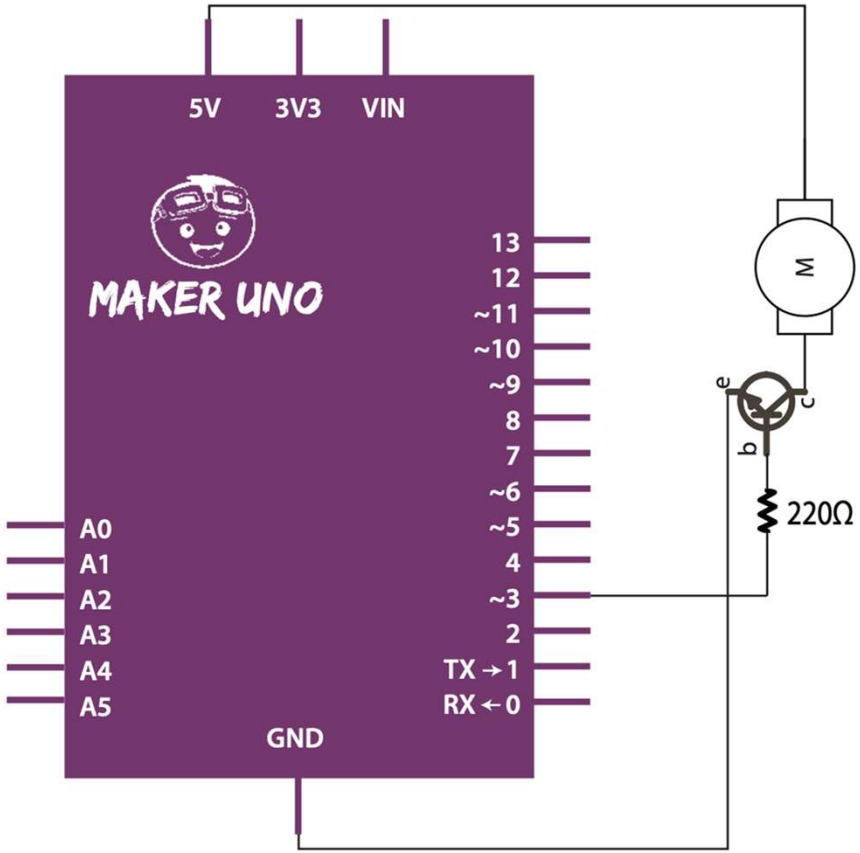
Controlling DC Motor

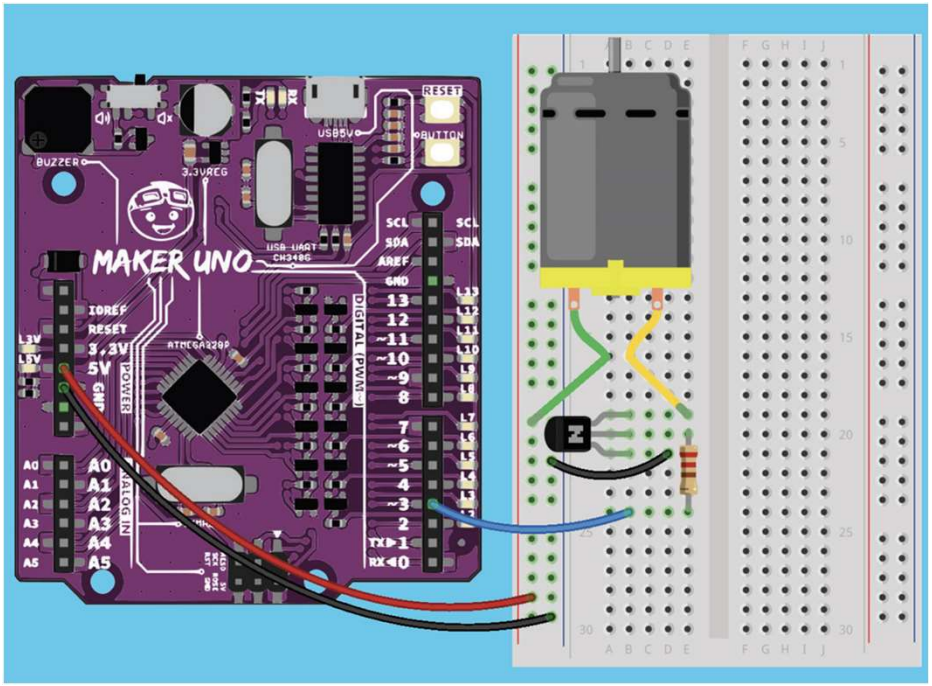
- We can control the speed of the DC motor by simply controlling the input voltage to the motor and the most common method of doing that is by using PWM signal
- Depending on the size of the motor, we can simply connect an Arduino PWM output to the base of transistor or the gate of a MOSFET and control the speed of the motor by controlling the PWM output.
- The low power Arduino PWM signal switches on and off the gate at the MOSFET through which the high power motor is driven



Activity: Controlling Motor

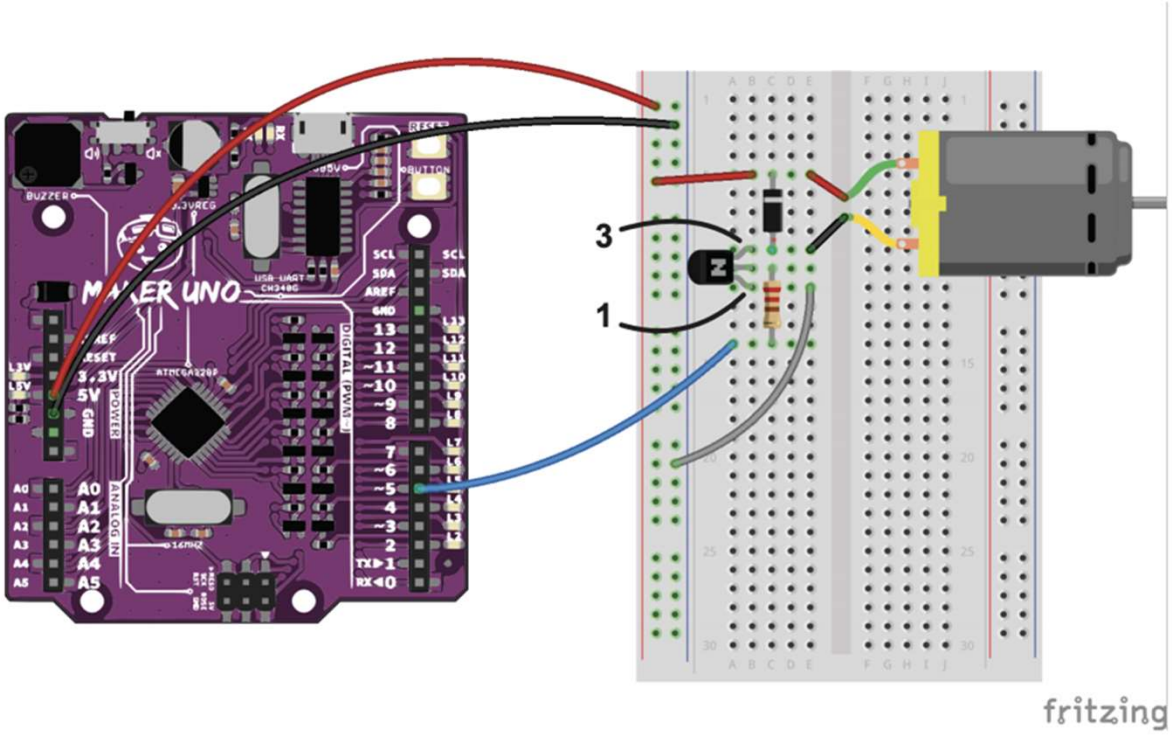
- Connect the circuit as shown below
- Write a sketch to control the DC motor



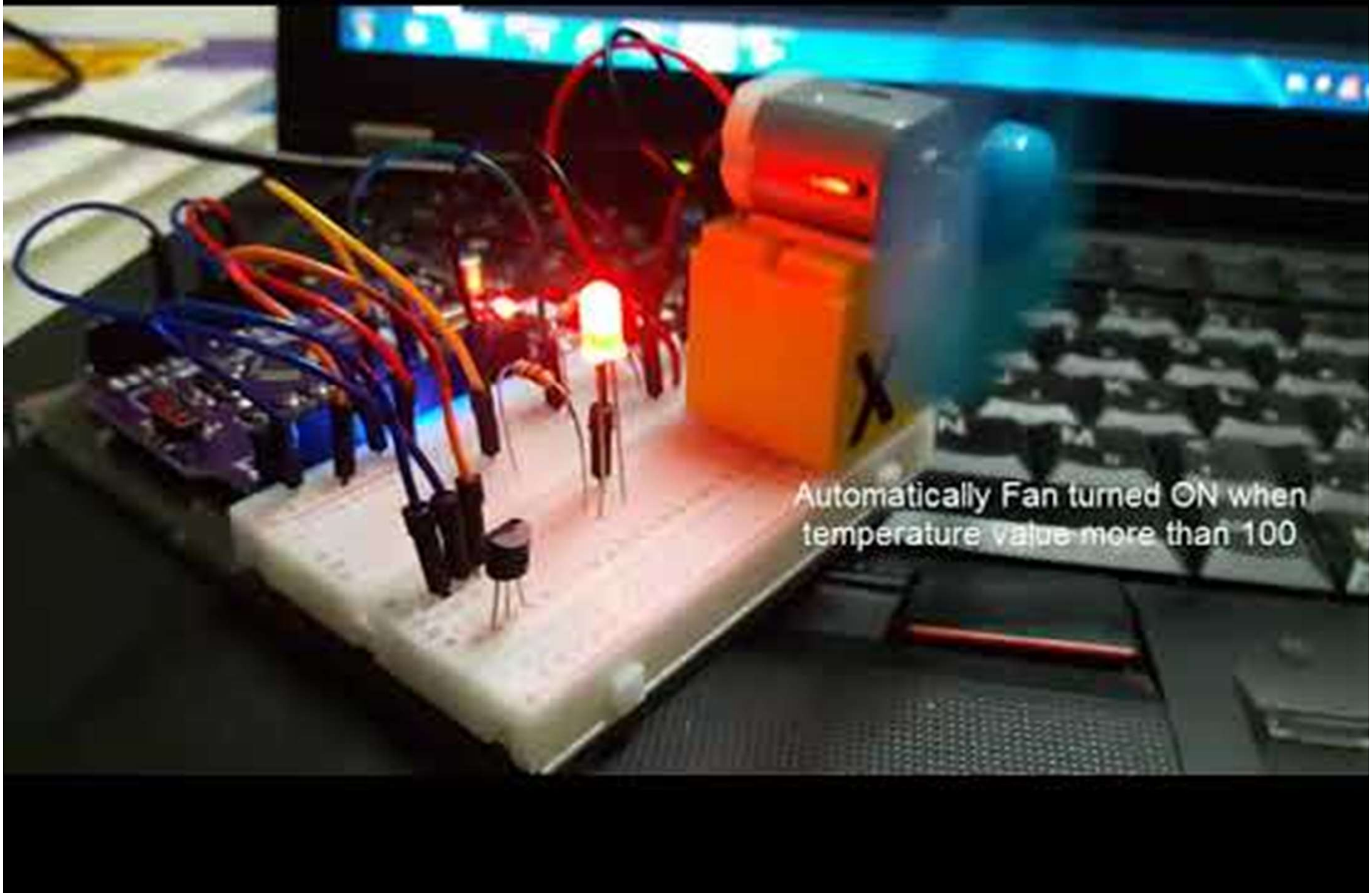


Activity: Controlling Motor

- Connect the circuit as shown below
- Write a sketch to control the DC motor

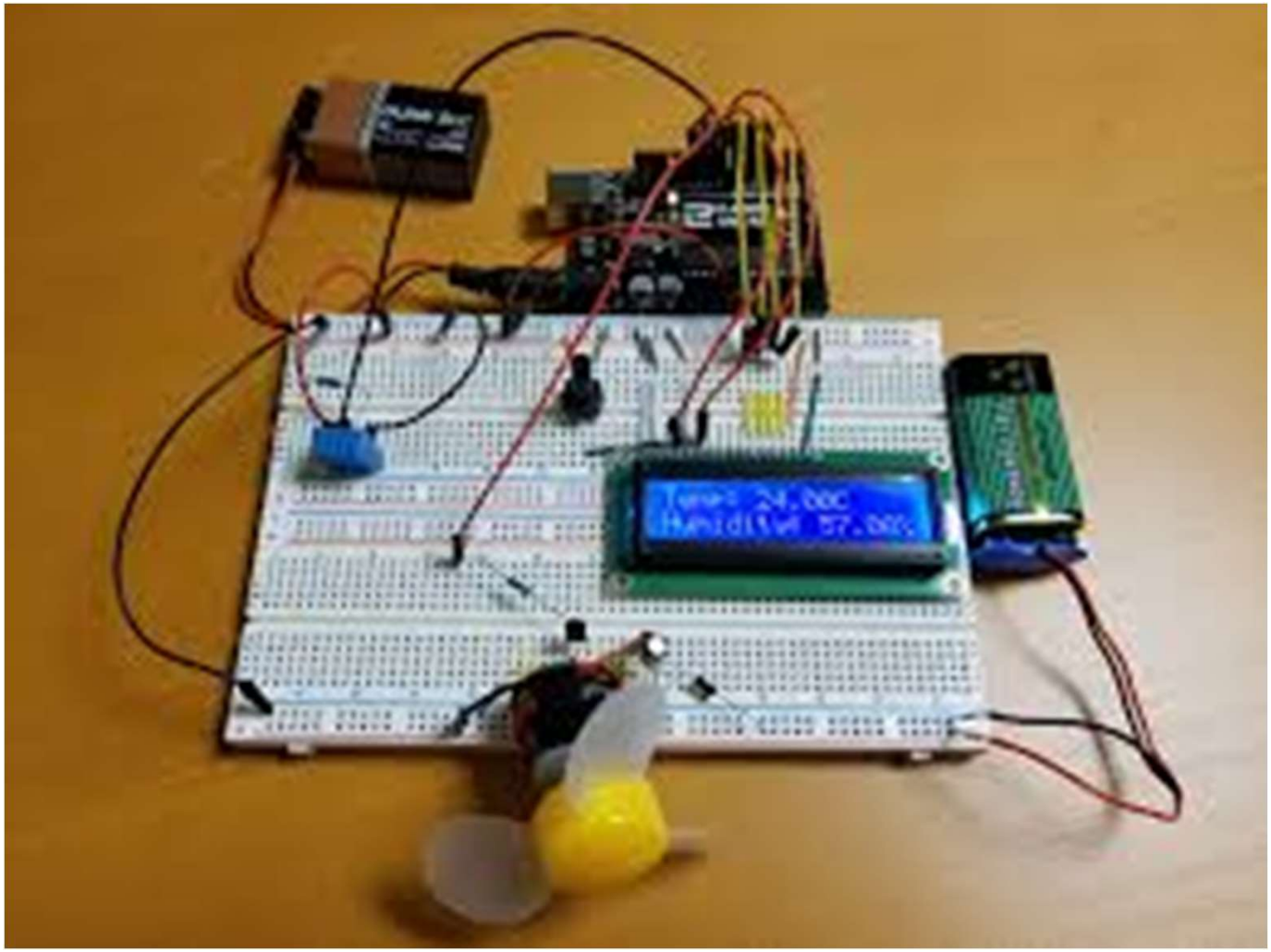


Fan Control System using Arduino



Activity: DC Fan Control with DHT

- Create a circuit and sketch to turn on the DC motor fan when the humidity is more than 50%



Summary
Q&A

Feedback

https://goo.gl/R2eumq

TINFOTECH

CERTIFICATE

of ACCOMPLISHMENT

You will receive a digital certificate in your email
after the completion of the class
If you did not receive the digital certificate,
please send your request to
enquiry@tertiaryinfotech.com

Thank You!

Shahul Maricar
shmaricar@gmail.com